UNIVERSITÉ DE MONTRÉAL

MIGRATIONS EN TEMPS RÉEL DES MACHINES VIRTUELLES INTERDÉPENDANTES

SALAH-EDDINE BENBRAHIM

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR

(GÉNIE INFORMATIQUE)

FÉVRIER 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

MIGRATIONS EN TEMPS RÉEL DES MACHINES VIRTUELLES INTERDÉPENDANTES

présentée par : BENBRAHIM Salah-Eddine

en vue de l'obtention du diplôme de : Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. PIERRE Samuel, Ph. D., président

M. QUINTERO Alejandro, Doctorat, membre et directeur de recherche

Mme BELLAICHE Martine, Ph. D., membre et codirectrice de recherche

M. CHAMBERLAND Steven Ph. D., membre

M. ST-HILAIRE Marc Ph. D., membre

DÉDICACE

À la meilleure des mamans, ma mère, R.E.M.

À ma merveilleuse épouse, L.K.

REMERCIEMENTS

Tout d'abord, j'aimerai remercier mon directeur de recherche, Monsieur Alejandro Quintero, ainsi que ma codirectrice de recherche, Madame Martine Bellaîche, qui sont des professeurs à l'École Polytechnique de Montréal et qui m'ont aidé dans les choix des volets de mon étude. Leur disponibilité et leur rigueur intellectuelle m'ont aidé à développer mes idées afin de résoudre la complexité de ce projet d'étude. Sans leur aide et leur appui scientifique, cette recherche n'aurait pas réussi ses objectifs. J'espère que cette collaboration pour ce sujet de recherche ne sera qu'un commencement d'un long chemin de partenariat pour des projets industriels et scientifiques.

De même, j'aimerai aussi remercier M. Samuel Pierre, professeur à l'École Polytechnique de Montréal, pour ses conseils qui m'ont été indispensables pour réussir une bonne méthodologie de recherche scientifique.

Je remercie aussi mes collègues du laboratoire de M. Quintero et du laboratoire LARIM pour les discussions scientifiques qui m'était utile pour être à jour sur mes connaissances scientifiques et techniques sur les sujets de réseaux, de télécommunication et des réseaux virtuels.

Merci aussi à M. Quintero pour son support financier me permettant de me consacrer pleinement à mes études.

Un grand merci pour le jury qui accepte de réviser et d'évaluer mon projet d'étude.

Merci aussi à ma mère et à mon épouse pour leur appui psychologique qui m'est indispensable pour surmonter les moments difficiles et de réussir mon projet d'étude.

RÉSUMÉ

Actuellement, les bonnes circulations et traitements des données sont devenus des clefs de succès dans tous les domaines techniques. Il est donc indispensable que les réseaux, véhiculant ces données, garantissent la qualité de leur transmission et réception. Cela est aussi applicable quand ces données sont échangées en continu par des hôtes virtuels distribués et interdépendants.

La consolidation et répartition des charges (Load Balancing) devient un élément important pour améliorer les capacités et les services des réseaux. Cette répartition est possible par des réseaux infonuagiques utilisant des machines virtuelles. Comme ces machines peuvent être déployées et migrées en temps réel et à grande échelle, elles peuvent offrir de très bonnes possibilités de répartition des charges par leurs migrations en temps réel. Ainsi, il est important pour les décideurs de ces répartitions de charges de disposer des techniques efficaces permettant de minimiser les coûts de maintenance et de qualité de ces migrations en temps réel et de maximiser les retours sur investissement de leurs déploiements.

Le problème de planification, de consolidation et de migration en temps réel des machines virtuelles (VMs) consiste à identifier les bons choix à effectuer pour placer les ressources d'un réseau de machines virtuelles et à déterminer les bonnes techniques pour les consolider par le déplacement de ces VMs entre des hôtes physiques. Ces déplacements des VMs doivent se faire sans interruption de service et dans des délais très réduits afin de respecter les contrats de niveaux de service et la qualité globale des services virtuels. Depuis quelques années, plusieurs recherches scientifiques se sont attardées sur l'étude de planification et de migration en temps réel des machines virtuelles. Cependant, ces études scientifiques se sont focalisées sur des VMs prises individuellement et non sur l'ensemble des VMs interdépendantes; cette démarche ne permet pas d'obtenir des solutions optimales prenant en considération les interdépendances entre ces VMs. D'autres études scientifiques se sont intéressées aux redéploiements dynamiques des charges d'un réseau en utilisant des migrations en temps réel des VMs et elles ont réussi à présenter des modélisations pour les résoudre. Cependant, elles n'ont pas considéré l'interdépendance entre des services applicatifs installés sur ces VMs; ces services ont besoin parfois d'échanger leurs informations afin d'effectuer leurs exécutions correctement.

Cette thèse présente des techniques traitant de la planification, la consolidation, et la migration en temps réel de plusieurs machines virtuelles. L'interdépendance entre les VMs est considérée lors

des développements de ces techniques. Notre travail est divisé en trois parties où chacune correspond à un de nos objectifs de recherche.

Lors de la première partie, nous développons un modèle mathématique et son heuristique d'approximation pour solutionner le problème d'optimisation de la planification des emplacements des VMs prenant en considération leurs contraintes d'interdépendance; cette heuristique, efficace pour des grands ensembles de machines virtuelles, peut être ensuite exécutée par un logiciel installé sur une machine physique. Nous résolvons ce modèle grâce au solveur mathématique CPLEX en utilisant la méthode de programmation en nombres entiers mixte (MIP). L'objectif de ce modèle est de minimiser les coûts d'un réseau de VMs tout en respectant ses contraintes d'interdépendance. Afin de valider la performance de notre modélisation, nous résolvons avec CPLEX des petits ensembles de tests; ensuite, nous les vérifions et validons. Notre modèle est plus pertinent que celui de la modélisation sans les contraintes d'interdépendance qui donne des solutions dans des délais plus courts, mais qui demeurent non efficaces pour le bon fonctionnement des VMs interdépendantes. Comme le problème NP-difficile de « bin-packing » peut être réduit à notre problème de planification des VMs, ce dernier est aussi NP-difficile; pour cela, nous approximons sa modélisation avec une heuristique de recherche taboue qui est capable de traiter des grands ensembles en peu de temps. Cette heuristique trouve de bonnes solutions dans des délais acceptables pour des problèmes avec des grands ensembles. Cette heuristique visite une zone de solutions potentielles afin d'y trouver l'optimum local, puis elle refait cette même démarche dans des régions avoisinantes. Ainsi, notre heuristique réalise une exploration pertinente de l'espace des solutions possibles. Les performances de notre heuristique sont comparables à celles de notre modèle mathématique approximé dans le cas des problèmes avec des petits ensembles. De plus, cette heuristique est plus performante en ce qui concerne des temps de calcul puisqu'elle réussit à trouver de bonnes solutions dans des délais moindres que ceux des solutions exactes de CPLEX pour des grands ensembles de VMs.

Lors de la deuxième partie de notre recherche, nous développons un modèle mathématique « multi-objectifs » (relaxé à un modèle « mono-objectif » par la méthode agrégée de la somme pondérée) et son heuristique d'approximation pour trouver une solution optimale pour le problème de migration en temps réel des VMs interdépendantes; cette modélisation obtient des solutions exactes et optimales pour un nombre réduit de VMs; cette heuristique, permettant de trouver des solutions quasi-optimales en peu de temps, peut être exécutée par un logiciel contrôleur installé sur une

machine physique. Cette exécution peut être effectuée à intervalle régulier ou bien quand la qualité de service de certains services virtuels commence à se dégrader. En effet, avec cette modélisation « multi-objectifs » relaxé à « mono-objectif » via la méthode agrégée de la somme pondérée de chacun de nos objectifs, nous trouvons des solutions quasi-optimales pour nos quatre objectifs qui sont le délai de migration des VMs, leur délai d'arrêt, les pénalités dues aux non-respects des contrats de service, et leur qualité de service globale. La modélisation proposée permet plus de flexibilité en assignant un niveau relatif d'importance pour chacun des objectifs via leur facteur de pondération. Concernant la qualité globale de service et les pénalités des non-respects des contrats de niveaux de service (SLAs), les résultats obtenus grâce à notre modèle et son heuristique d'approximation sont meilleurs que la technique « pré-copie » utilisée et conçue pour les migrations en temps réel des VMs.

Lors de la troisième partie de notre recherche, nous développons un modèle mathématique et son heuristique d'approximation visant à maximiser le profit net total tout en minimisant les pénalités des services virtuels aux contrats SLAs. Cette optimisation est une tâche complexe en raison de la difficulté de parvenir à un compromis réussi entre les pénalités sur les contrats de niveaux de service et le placement en temps réel des machines virtuelles (VM) interdépendantes. Cette troisième partie étudie donc ce problème de maximisation du profit net total tout en diminuant les pénalités de service et en réalisant des migrations en temps réel de machines virtuelles interdépendantes. Ce problème d'optimisation de placement en temps réel de machines virtuelles est NP-difficile puisque le problème NP-difficile « bin-packing » peut être réduit à ce problème, et son temps de calcul croit exponentiellement avec la taille des ensembles de machines virtuelles et de leurs machines physiques hôtes; pour cette raison, nous approximons notre modèle mathématique avec une heuristique de recherche taboue efficace. Nous testons notre formulation et heuristique pour des services virtuels, où le profit net total doit être maximisé, où les pénalités des services doivent être minimisées, et où des migrations efficaces en temps réel des VMs sont des sujets de préoccupation. Nos résultats de simulations montrent que notre heuristique d'approximation de notre modèle mathématique : (i) trouve de meilleures solutions que la configuration existante des milliers de machines utilisées dans des traces de Google; (ii) est adaptée pour de grandes ensembles des services virtuels avec des dizaines de milliers de machines virtuelles et machines physiques; et (iii) performe mieux en termes de pénalités et profits globaux que notre

référence de comparaison c.-à-d. la configuration existante des machines utilisées dans des traces de Google.

ABSTRACT

Currently, data transmission and processing have become keys to success in many technology areas. Therefore, it is essential that networks guarantee the transmission and reception qualities of these data; also, this guarantee is important for data exchanged continuously by distributed and interconnected hosts.

Also, "Load Balancing" techniques becomes an interesting key to improve network services and capacities. This load balancing technique is feasible with cloud networks based on virtual machines (VMs). Since these virtual machines can be deployed and live migrated on a large scale, they can offer very good possibilities of load balancing. Thus, it is important that decision makers dispose of effective techniques, such as load balancing, to minimize costs of these live migrations and to maximize their return on investment.

Planning and live migration problems of virtual machines aim to identify right choices to place resources of virtual machine networks and to determine right techniques to move VMs between their physical hosts. These VMs' moves should be done without service interruption and within very short delay. In recent years, several scientific researchers have studied planning and live migration of virtual machines; however, these scientific studies have focused on VMs individually without considering their interdependency constraints. Other scientific studies have investigated dynamic load balancing of a network using VM live migrations and they have succeeded to solve it using mathematical models; however, they did not consider the VMs' interdependency.

This thesis presents some techniques dealing with planning, consolidation, and live migrations of multiple virtual machines. These techniques take into account the VM interdependencies, the network service level contracts (SLAs) and overall quality. Our thesis is divided into three parts corresponding to our three research objectives.

In the first part, we develop a mathematical model for VMs planning problem including the interdependency constraints. We solve this model using CPLEX as a mathematical solver with the mixed integer programming (MIP) method. The goal of this model is to minimize the overall cost of a VMs' network while respecting its interdependent VMs constraints. To validate our modeling performance, we solved, with CPLEX, some small sets; then, we verified and validated our solutions. Our model is more relevant than other models ignoring interdependency constraints and which give solutions in a shorter time but remain ineffective for a proper functioning of

interdependent VMs. Since VMs' placement planning problem is an NP-hard problem (as the NP-hard "bin-packing" problem can be reduced to it), we approximate our model with a tabu search heuristic which is capable to handle large-sized sets. This heuristic finds good solutions in an acceptable delay by visiting areas of potential solutions in order to find local optimums, and then it repeats this process with other surrounding areas; thus, our heuristic performs relevant space explorations for possible solutions. Our heuristic has comparable performance to our approximated mathematical model in the case of small size sets; moreover, this heuristic is more efficient since its running time is lower than CPLEX to find near-enough exact solutions for large-sized sets.

In the second part of our research, we have developed a "multi-objective" mathematical model to solve the problem of live migrations of interdependent VMs; this model is solved by relaxing it to a "mono-objective" model using the method of weighted sum of each of its objectives; thanks to this relaxation, we optimize four objectives simultaneously by dealing with live VM migrations, especially, their stop delays, their penalties on service level agreements and the overall quality of service. Our proposed model allows more flexibility by assigning a relative importance level for each objective. Results, of the overall quality of service and overall penalty on SLAs, obtained with our model are better than those of "pre-copy" VM live migrations ignoring VMs' interdependency constraints.

In the third part of this research, we develop a mathematical model to maximize the overall net profits of virtual services while minimizing the penalties on their SLAs. This optimization is a complex task because it is difficult to reach a successful compromise between decreasing the penalties on service level agreements and the live placement of interdependent virtual machines. This third part is therefore exploring this problem optimization of net profit and service penalties while performing live migrations of interdependent virtual machines. This live placement optimization problem of virtual machines is NP-hard and its calculation time grows exponentially with the size of virtual machine sets and their physical machines hosts; for this reason, we approximate our mathematical model with effective tabu search heuristic. We test our formulation and heuristic with virtual services which the overall net profit should be maximized, the penalty on SLAs should be minimized and live migrations should be effective. Our simulation results show that our heuristic: (i) finds better solutions than the existing configuration of machines used in Google traces; (ii) is adapted to large-sized virtual service sets with tens of thousands of virtual and

physical machines; and (iii) performs better in terms of penalties and overall net profits when they are compared to those of the existing machines' configuration of Google trac

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABREVIATIONS

Ajax	Appels asynchrones des fonctionnalités de JavaScript et de XML (Asynchronous JavaScript XML).
API	Interface de programmation des applications (Application Interface Programming).
ARP	Protocole de résolution des adresses (Address Resolution Protocol).
DASM	Machine d'états abstraite distribuée (Distributed Abstract State Machine).
DHT	Table de hachage distribuée (Distributed Hash Table).
DR	Recouvrement de pannes (Disaster Recovery).
FC	Protocole de connexion à haut débit (Fibre Channel).
FCAPS	Un modèle de gestion des réseaux (Fault, Configuration, Accounting, Performance, Security).
FF	Algorithme de placement au premier emplacement suffisant (First Fit).
FFD	Algorithme de tri descendant suivi par du placement au premier emplacement suffisant (First Fit Decreasing).
GA	Algorithme génétique (Genetic Algorithm).
GB	Quantité en Gigaoctets (Giga Bytes).
GPV	Problème d'optimisation des planifications des machines virtuelles (problem of Global Planning of Virtual networks).
HA	Disponibilité élevée (High Disponibility).
HAV	Matériel d'assistance à la virtualisation (Hardware-Assisted Virtualization).

HPL	Implémentation des tests de calcul mathématique (High Performance Linpack).
HR	Routeur des données des disques durs (Hard disk Routers).
IaaS	Infrastructure comme service (Infrastructure as a Service).
IOPS	Opérations d'entrées/sorties par seconde (I/O operations Per Second).
IP	Protocole d'Internet (Internet Protocol).
IPOP	Un système de création de réseaux IP virtuels sur des réseaux distribués (IP Over P2P).
JSON	Représentation textuelle des objets (JavaScript Object Notation).
JSP	Page web d'un serveur Java (Java Server Page).
KVM	Machine virtuelle intégrée au noyau du système d'exploitation (Kernel-based Virtual Machine).
LAN	Réseau local (Local Area Network).
LM	Migration en temps réel (Live Migration).
MAC	Sous-couche de contrôle d'accès aux médias (Media Access Control).
MCDA	Approche d'analyse de décision multicritères (Multiple Criteria Decision Analysis).
MOGA	Algorithme génétique « multi objectifs » (Multiple Objective Genetic Algorithm).
MP	Machine physique.
MPI	Interface de passage des messages des applications avec mémoires distribuées (Message Passing Interface).

MPICH2	Implémentation populaire de l'interface de passage des messages (MPI).
MR	Routeur des données des mémoires (Memory Routers).
MTTF	Temps moyen d'une panne (Mean Time To Failure).
NAT	Traduction d'adresses réseau (Network Address Translation).
NP	Non polynomial.
NRBC	Menace nucléaire, radiologique, biologique et chimique (Nuclear Radiological Biological Chemical).
NSGA	Algorithme génétique trié et non dominé (Nondominated Sorting Genetic Algorithm).
OMEN	Simulateur de nanoélectronique quantique.
OS	Système d'exploitation (Operating System).
PaaS	Plateforme comme service (Platform as a Service).
PCB	Bloc du contrôle des protocoles (Protocol Control Block).
PCViMaLMi	Migration en temps réel des machines virtuelles parallèles et coopératives (Parallel Cooperative Virtual Machine Live Migration).
PCVM	Machines virtuelles parallèles et interdépendantes (Parallel Cooperative Virtual Machine).
PDN	Réseau public de données (Public Data Network).
PM	Machine physique (Physical Machine).
PVM	Machine virtuelle parallèle (Parallel Virtual Machine).
QdS	Qualité de service (Quality of Service).

QoS	Qualité de service (Quality of Service).
SaaS	Logiciels comme service (Software as a Service).
SAN	Réseau haut débit de stockage (Fibre Channel-based Storage Area Networks).
SD	Algorithme de distribution simple (Simple Distributed).
SDM	Méthode de distribution simple (Simple Distributed Method).
SDSRV	Serveur des données partagées et des disques durs virtuels (Shared Data and virtual hard disk Servers).
SLA	Contrat du niveau de service (Service Level Agreement).
SLC	Vérification du niveau de service (Service Level Checking).
SLO	Objectif du niveau de service (Service Level Objectif).
SOA	Architecture orientée service (Service Oriented Architecture).
SPEA	Algorithme d'évolution forte de Pareto (Strength Pareto Evolutionary Algorithm).
TCP	Protocole du contrôle des transmissions.
TI	Technologies d'informations.
TS	Heuristique de recherche taboue (Tabu Search).
VEGA	Algorithme génétique d'évaluation par vecteurs (Vector Evaluated Genetic Algorithm).
vHD	Disque dur virtuel (virtual Hard-Disk).

ViMaLMi	Migration en temps réel des machines virtuelles (Virtual Machine Live Migration)
VM	Machine virtuelle (Virtual Machine).
VMM	Gestionnaire des machines virtuelles (Virtual Machine Manager).
VNS	Heuristique de recherche à voisinage variable (Variable Network Search).
VPC	Infonuagique virtuellement privé (Virtual Private Cloud).
VPLS	Service du réseau local virtuel privé (Virtual Private LAN Service).
VPN	Réseau virtuel privé (Virtual Private Network).
WAN	Réseau étendu (Wide Area Network).
WWS	Pages mémoires utilisées pendant un certain intervalle de temps (Writable Working Set).

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Ce chapitre a pour objectif de présenter nos recherches sur les machines virtuelles (VMs) dans le cadre de la planification, consolidation, et migration de plusieurs machines virtuelles. Dans les sections suivantes, nous présentons les concepts de base du domaine des réseaux virtuels et leurs machines virtuelles, ainsi que notre problématique, et nos objectifs de recherche.

Il y a quelques années, la migration des machines virtuelles ne se faisait pas en temps réel, mais elle se réalisait seulement après un arrêt complet des VMs. Cela pouvait s'expliquer par le manque d'outils automatiques pour une migration en temps réel (LM — Live Migration) des VMs. Cependant, en réponse à la croissance marquée du nombre des VMs et des réseaux virtuels, des outils automatisés de migration en temps réel des VMs sont devenus indispensables. De plus, vu le contexte économique actuel, les fournisseurs des services de réseaux des VMs ont avantage à effectuer des migrations rigoureuses pour fidéliser leurs clients en leur offrant des services continus meilleurs que leurs concurrents leur permettant ainsi de gagner plus de parts de marché.

Les réseaux virtuels et les VMs sont de plus en plus populaires et le nombre de leurs utilisateurs ne cesse d'augmenter. Par conséquent, les fournisseurs de ces services n'ont d'autres choix que d'augmenter leurs investissements afin de répondre efficacement aux besoins grandissants de leurs clientèles. Tout comme la migration en temps réel d'une machine virtuelle dans des réseaux LANs, de bonnes techniques sont aussi nécessaires pour la migration simultanée en temps réel totale ou partielle de plusieurs VMs coopératives ou non dans les différents types de réseaux comme les LANs et les WANs. Ces techniques ont l'objectif de minimiser les coûts des délais d'arrêt des services dus à la migration simultanée de plusieurs VMs tout en respectant certaines contraintes comme la qualité de service; ainsi, les fournisseurs des services de virtualisation peuvent réduire leurs coûts de planification et de maintenance pour faire face à la dégradation éventuelle de la qualité de leurs services. Cependant, toute migration en temps réel des VMs doit permettre la continuité des fonctionnements des services de ces VMs lors de leurs migrations, en diminuant le temps total de migration et des temps d'arrêt des services hébergés sur des machines virtuelles.

Dans cette thèse, nous nous intéressons à la planification, la consolidation, et la migration en temps réel globale et simultanée de plusieurs VMs. Ainsi, ce chapitre d'introduction discute des avantages économiques affirmant la valeur ajoutée des sujets de notre étude et de la situation actuelle du sujet

de la migration des VMs. Ensuite, nous définissons sommairement les concepts de base, la problématique, et les trois objectifs de recherche de cette étude. Enfin, nous présentons les chapitres suivants de ce document.

1.1 Situation actuelle des migrations en temps réel des VMs et leurs avantages

1.1.1 Avantages économiques et financiers

Du point de vue économique et financier, cette migration en temps réel des VMs peut avoir des retombées économiques positives pour les fournisseurs et les responsables des services informatiques, car elle permet de générer des flux monétaires supplémentaires dus aux économies d'échelle résultant de la création et l'hébergement simultanés de plusieurs VMs sur une ou plusieurs machines physiques plus performantes; de plus, elle permet de meilleures répartitions de charge (load balancing) en migrant des VMs (hébergées sur des machines physiques hôtes peu performants) vers de nouveaux hôtes plus performants. Ainsi, une entreprise peut diminuer ses coûts d'immobilisations corporelles, augmenter son chiffre d'affaires grâce à l'amélioration de la qualité de ses services, augmentant ainsi sa marge bénéficiaire nette par rapport à ses ventes brutes. Les coûts d'exploitation, liés aux coûts d'immobilisations corporelles dans les divisions des services informatiques des entreprises, représentent généralement une grande part du budget du fonctionnement d'une entreprise, et ils sont dus aux loyers, achats et aux amortissements des équipements informatiques physiques et logiciels, aux loyers des locaux de ces machines physiques, etc. Une partie de ces coûts d'exploitation peut être diminuée α fois (où $\alpha > 2$) grâce à l'utilisation de « n VMs hébergées sur p machines physiques » où « n très grand que p », ce qui peut diminuer les coûts dispendieux des recours au déploiement de n machines physiques (PMs). Dans le même sens, il est important de souligner aussi que la migration simultanée de plusieurs VMs permet aux décideurs, stratégistes des systèmes d'information dans une entreprise, d'avoir en main les clefs nécessaires pour choisir et modifier librement les lieux physiques de déploiement de leurs services et de changer leurs fournisseurs des services informatiques dans des délais raisonnables en exécutant tout simplement des migrations en temps réel des VMs d'un lieu ou d'un fournisseur vers d'autres horizons plus opportuns économiquement et techniquement. Évidemment, l'importance de ces aspects est proportionnelle au nombre des VMs à migrer.

1.1.2 Situation actuelle

Actuellement, il existe un grand intérêt scientifique et économique pour l'utilisation des machines virtuelles. Cependant, d'après notre constat, il n'y a pas beaucoup de recherches sur la migration partielle ou totale en temps réel et simultanée de plusieurs machines virtuelles d'un domaine donné vers un autre domaine, tout en supportant la continuité des calculs parallèles et dépendants de ces machines virtuelles migrées en temps réel. Les recherches actuelles se focalisent surtout sur la migration individuelle des VMs interdépendantes dans un réseau LAN ou VPN [1]. De même, Xu et al. [2] confirment la difficulté d'utiliser les techniques actuelles de migration en temps réel (Live Migration – LM) dans les réseaux WANs [2] à cause des problèmes de connexion bidirectionnelle dus aux NATs [2].

Les techniques utilisées actuellement (dite approche classique ou non-globale) pour maintenir les services des VMs individuelles ne sont pas utilisables comme telles pour les migrations en temps réel de plusieurs machines virtuelles parallèles et interdépendantes à cause de l'interdépendance des VMs migrées, des problèmes de congestion de réseaux et à cause des problèmes d'adressage liés aux réseaux WANs. Ainsi, nous constatons que ce type de migration est un problème très complexe à résoudre.

Plusieurs moyens d'interaction existent actuellement entre différentes VMs coopératives. Par exemple, une VM peut utiliser une mémoire partagée pour communiquer avec une autre VM, ce qui implique que la migration d'une de ces deux VMs coopératives a assurément un impact sur l'autre VMs partenaires en ce qui concerne la préparation des données pour leur transfert, leurs transferts et leur réactivation sur de nouvelles machines physiques (PMs) hôtes. Selon notre constat, les techniques actuelles traitent la migration d'une VM indépendamment des autres VMs; ainsi, les différentes interactions entre différentes VMs ne sont malheureusement pas prises en considération. De plus, selon notre revue de littérature scientifique, aucune stratégie n'est définie actuellement pour la migration simultanée des ressources partagées, ce qui nous laisse avec des VMs optimisées pour une migration individuelle et non pour une migration groupée optimisée (c.-à-d. une migration globale).

Autrement dit, actuellement, la synchronisation des fonctionnements des VMs coopératives et parallèles, avant, pendant, et après une migration partielle ou totale, est fragile, difficile à réaliser

dans un temps raisonnable; de plus, elle peut ne pas fonctionner correctement à cause des NATs/Firewalls et de la congestion; par conséquent, cette migration doit être traitée avec précaution. Ainsi, Romero et al. [3] [4] mettent en évidence que la migration simultanée de plusieurs VMs (avec un exemple de huit VMs) retarde l'exécution de l'application HPL jusqu'à 54 fois.

1.2 Définitions et concepts de base

Dans cette section, nous détaillons quelques concepts utilisés au cours de cette thèse.

1.2.1 Répartition des charges (Load Balancing)

La répartition de charges (load balancing) est la technique permettant de distribuer les charges entre plusieurs machines faisant partie d'un groupe à l'insu de l'usager. Ainsi, cette technique permet d'obtenir des dispositifs très puissants, de réduire les temps de réponse, et d'augmenter l'évolutivité et la disponibilité d'un serveur informatique. Le domaine d'étude de la « répartition des charges » est issu de la recherche dans le domaine des ordinateurs parallèles. Actuellement, cette « répartition de charges » est construite en utilisant une collection de machines similaires souvent appelée « cluster » ou « ferme de serveurs » et en distribuant les efforts entre ces machines. Dans ce contexte, les demandes de service (comme les consultations d'une page web) sont dirigées vers un routeur plutôt qu'un ordinateur permettant ainsi de distribuer à tour de rôle le traitement de ces services entre les ordinateurs d'un cluster selon des degrés de priorités.

1.2.2 Infonuagiques (Cloud Computing)

- **Définition d'infonuagique**

Une des définitions les plus utilisées pour l'infonuagique est celle de L. M. Vaquero [5] où l'auteur affirme que les infonuagiques sont un grand ensemble de ressources virtuelles (comme du matériel informatique, des plateformes et/ou des services) accessibles et facilement utilisables. Ces ressources peuvent être dynamiquement reconfigurables afin de s'ajuster à la variabilité de la taille d'un réseau virtuel et permettant une utilisation optimale des ressources. Cette réserve de ressources est typiquement exploitée par un « paiement à l'utilisation » avec une garantie de la part des fournisseurs du respect des SLAs (Service Level Agreements) personnalisés.

L'objectif des réseaux virtuels est donc l'optimisation de l'utilisation des ressources physiques et logicielles, l'amélioration de leur flexibilité, et de l'automatisation de leur gestion tout en externalisant les réserves en ressources auprès des tierces parties [6].

Cependant, plusieurs problématiques, soulevées par le réseau virtuel, liées en grande partie à l'implémentation des SOAs (architecture orientée services) et de la virtualisation matérielle et logicielle. De même, l'infonuagique soulève d'autres problématiques liées aux contrats de service, leur gestion, et leur vérification [6].

- **Parties prenantes des infonuagiques**

Selon Vijay et al. [7], une plateforme générique des services virtuels doit répondre aux besoins des parties prenantes suivantes :

i. **les fournisseurs de l'infrastructure de réseaux virtuels** : sont des fournisseurs des ressources de calcul, de réseaux et de stockage. Ces ressources peuvent être utilisées pour créer des ordinateurs virtuels logiques permettant la livraison des services modulables et portables. Cette infrastructure est utilisable à la fois par les développeurs informatiques et les utilisateurs de ces services. Le modèle proposé par ces auteurs ressemble à celui utilisé par les fournisseurs des produits et des services de télécommunications (« switching, transmission and Access Equipment, service Ealing factures and management interfaces in Equipment »). L'infrastructure actuelle de stockage et de calcul manque de gestion dynamique permettant d'éliminer la « latence humaine de la gestion »;

ii. **les fournisseurs des services** : offrent des services à la demande et aussi des indicateurs sur l'utilisation des ressources fournies afin de gérer au moment opportun les spécifications de leur disponibilité et de leur sécurité; ainsi, il est possible de répondre aux SLAs et de créer de nouveaux systèmes d'exploitation respectant des contraintes des applications virtuelles;

iii. **les développeurs des services** : offrent des services virtuels en utilisant des interfaces de programmation de ces services afin de configurer, surveiller et gérer des ressources en ce qui concerne leurs allocations, leurs disponibilités, leurs utilisations, leurs performances, et leur sécurité;

iv. **les utilisateurs finaux** : font des demandes croissantes afin d'avoir les moyens nécessaires pour une meilleure interactivité avec les ressources virtuelles à travers des interfaces intuitives. Ces utilisateurs finaux peuvent être amenés à payer selon leur consommation des ressources correspondantes aux SLAs et qui sont calculées à l'aide de certaines mesures; ces mesures sont intégrées lors de la phase de développement des services et mises à la disposition des fournisseurs des services.

Ayant présenté quelques concepts de base du domaine des infonuagiques et des machines virtuelles, nous présentons, dans la section suivante, notre problématique et les objectifs de notre étude.

1.3 Éléments de la problématique

L'explosion de l'utilisation des réseaux virtuels amène de nouvelles problématiques à leurs fournisseurs en ce qui concerne la planification des machines virtuelles, de leurs ressources distribuées, de leur consolidation, et de leurs migrations en temps réel.

La planification et la consolidation des VMs sont des problèmes NP-difficiles à résoudre [8] car le problème NP-difficile « bin-packing » [9] peut leur être réduit. L'étape de définition des emplacements des VMs est importante pour trouver les meilleurs emplacements de ces VMs permettant de diminuer les coûts de location et/ou d'achat des ressources physiques et des coûts des ressources humaines liées aux délais d'installation et de replacement de ces VMs à l'aide de migration en temps réel. Théoriquement, il est possible de trouver ces emplacements idéaux des VMs en choisissant les meilleurs de tous les emplacements possibles; cependant, dans la pratique, il est très difficile de faire le tour de toutes les possibilités de placement de VMs dans des délais raisonnables.

Notre étude traite la planification, la consolidation, et la migration simultanée de plusieurs machines virtuelles interdépendantes. Cette planification, consolidation, et migration de VMs sont indispensables pour les fournisseurs des services des infonuagiques car elles leur permettent par exemple de réduire leurs coûts d'installation et maintenance des réseaux de VMs; elles permettent aussi de faire migrer et consolider des VMs parallèles hébergées sur des plateformes physiques de moyennes capacités vers d'autres plateformes physiques plus performantes. De plus, une migration

en temps réel supporte plusieurs techniques comme la tolérance aux pannes (fault tolerance) et la répartition efficace des ressources (load balancing).

Ce type de migration de VMs est peu traité dans la littérature scientifique; d'après nos recherches, nous ne trouvons généralement dans la littérature scientifique que des études d'amélioration de la migration d'une seule machine virtuelle et non la migration simultanée de plusieurs machines virtuelles interdépendantes; la migration groupée doit prendre en considération simultanément les différents moments de transfert, sur le réseau, les différentes quantités des copies des états de chacune des VMs, les ressources physiques des débits du réseau, et les capacités physiques des machines physiques hôtes. Les études scientifiques existantes ne traitent pas la migration groupée optimisée de l'ensemble des machines virtuelles alors que cette migration groupée est exponentiellement influencée par le nombre de VMs migrées comme indiqué par Romero et al. [3] [4]; ces auteurs confirment que la migration simultanée, par exemple de huit VMs parallèles, est plus lente (jusqu'à 54 fois) que celle d'une seule VM. De même, les techniques décrites dans la littérature scientifique actuelle ne sont pas performantes pour la migration simultanée de plusieurs VMs parallèles et coopératives alors que ces VMs doivent rester connectées entre elles, lors de leurs migrations, pour faire fonctionner continuellement leurs services parallèles.

Suite à ce constat de manquement d'outils pour la planification et consolidation de VMs interdépendantes à l'aide de migration en temps réel, nous nous intéressons à la possibilité de trouver des solutions dont les objectifs généraux sont les optimisations des placements de VMs interdépendantes permettant la continuité des fluidités des services de ces VMs, la minimisation des temps d'arrêt de leurs services, la minimisation du temps total de leur migration, la diminution des temps de leur maintenance, la diminution de la congestion des réseaux, la consistance des mémoires des VMs, la continuité des connexions réseaux, et la consistance des états des applications même celles distribuées entre plusieurs processus. Par exemple, nous nous intéressons au bon fonctionnement d'un site web en ligne ou des serveurs de streaming de médias qui doivent continuer à fournir leurs services en tout temps sans avoir à demander à leurs clients de se reconnecter aux applications lors des pannes ou de transfert de leurs hôtes virtuels. Selon Christopher et al. [10], ceci est possible par l'utilisation de plusieurs techniques comme les migrations en temps réel des services virtuels et par, entre autres, des solutions des redirections des connexions de la couche 7. De plus, ces clients, des services virtuels, s'attendent à avoir

continuellement une bonne qualité de service lors des migrations des machines virtuelles entre des machines physiques hôtes.

D'un point de vue technologique, le problème réside dans la manière et le moment où des VMs (éventuellement interdépendantes), hébergeant des services connectés, peuvent se déplacer entre des hôtes physiques. Ces questions sont importantes pour les fournisseurs des services des centres virtuels de données qui veulent une bonne qualité et un meilleur rendement de leurs centres de données. Pour ces raisons, un réseau de machines virtuelles doit avoir une planification initiale des emplacements de ses VMs et une bonne redistribution de leurs charges permettant de garantir une bonne qualité de service de leur réseau. L'avantage, pour les fournisseurs des services virtuels, est de garantir les qualités de service et de respecter les contrats de niveaux de service tout en optimisant leurs coûts; cet avantage peut se réaliser par des optimisations des services aussi bien à la demande initiale qu'aux moments de redistribution des services.

Ainsi, à partir de la problématique décrite ci-haut, nous avons recensé les trois questions suivantes, aboutissant aux trois objectifs de notre recherche, chacun d'eux faisant l'objet d'un article pour notre thèse :

i. comment définir une planification de machines virtuelles interdépendantes afin de s'assurer que leurs ressources virtuelles sont utilisées d'une manière optimale?
ii. comment assurer, lors des migrations en temps réel des machines virtuelles interdépendantes d'un réseau, sa qualité de service globale et le respect de ses contrats de service afin de réussir une bonne continuité et fluidité de ses services virtuels?
iii. comment assurer une consolidation de machines virtuelles interdépendantes tout en respectant leurs contraintes d'interdépendance, diminuant la pénalité globale sur ses contrats de service, et maintenant la qualité de service globale?

Après avoir annoncé, dans ce chapitre d'introduction, le problème général discuté dans cette étude, sa situation actuelle et notre problématique, nous présentons dans la section suivante nos objectifs.

1.4 Objectifs de recherche

L'objectif de cette thèse est de proposer de nouveaux modèles aux réseaux de machines virtuelles interdépendantes, en ce qui concerne la planification de leurs emplacements initiaux, leurs

consolidations et leurs migrations en temps réel entre des hôtes physiques. Ces modèles ont comme objectifs de respecter les contraintes SLAs d'un réseau de machines virtuelles et de maintenir sa qualité de service. Plus spécifiquement, cette thèse vise à :

i. proposer un modèle de planification d'un réseau de machines virtuelles interdépendantes pour minimiser le coût d'installation lié à leurs machines physiques et respecter les différentes contraintes entre ses VMs interdépendantes et de leurs ressources partagées. Ce modèle est ensuite testé, validé, et comparé à d'autres modèles existants dans la littérature scientifique;
ii. proposer un modèle mathématique pour améliorer la qualité globale des services et diminuer les temps de migration en temps réel des VMs interdépendantes. Ce modèle respecte les contraintes d'interdépendance et les SLAs. Ce modèle est ensuite testé, validé, et comparé à d'autres modèles existants dans la littérature scientifique;
iii. proposer un modèle mathématique et heuristique d'approximation pour des migrations groupées en temps réel de VMs interdépendantes afin de permettre une consolidation de leurs services tout en maintenant leur qualité de service, respectant leurs contrats de niveaux de service et leurs contraintes d'interdépendance. Ce modèle est ensuite testé, validé, et comparé à une configuration existante dans des centres de données virtuels.

Dans cette section, nous avons décrit les objectifs de notre étude; pour ensuite décrire, dans la section prochaine, le plan de la thèse.

1.5 Plan de la thèse

Lors de ce chapitre d'introduction, nous avons exposé quelques concepts de base nécessaires pour la compréhension du sujet couvert par cette étude, le problème général discuté dans cette étude et sa situation actuelle, notre problématique, et nos objectifs. Dans le deuxième chapitre, nous dressons une revue de littérature sélective de la planification, la consolidation, et la migration en temps réel d'une ou plusieurs VMs parallèles (coopératives ou non coopératives). Par la suite dans le troisième chapitre, nous présentons les démarches de l'ensemble du travail de recherche, alors que du quatrième au sixième chapitre, nous présentons nos deux articles scientifiques de conférences publiés et le troisième article soumis pour publication dans un journal. Lors du septième chapitre, nous effectuons une discussion générale de la démarche et des résultats de notre

étude. Enfin, dans le huitième chapitre, nous présentons la conclusion de cette thèse détaillant l'originalité de notre étude et ses principales contributions pour la recherche scientifique, sans oublier d'annoncer les limites de nos travaux actuels et nos éventuels travaux futurs.

CHAPITRE 2 REVUE CRITIQUE DE LA LITTÉRATURE

Au cours de ce chapitre, nous effectuons une revue des travaux scientifiques de la planification, consolidation, et migration en temps réel de machines virtuelles; ceci nous permet de présenter une analyse sommaire des problèmes des VMs. Ensuite, ce chapitre présente les différentes techniques utilisées actuellement pour résoudre ces problèmes. Enfin, nous présentons dans ce chapitre une analyse brève de ces travaux et techniques.

2.1 Analyse sommaire du problème

Les techniques de planification et migration de plusieurs machines virtuelles sont très similaires à celles d'une seule VM. Les problèmes de planification, consolidation, et migration de plusieurs VMs sont généralement divisés en plusieurs sous-problèmes traités individuellement. Cependant, il nous semble plus opportun de traiter simultanément et globalement ces sous-problèmes en les modélisant avec un même modèle mathématique; ce traitement simultané permet d'avoir de meilleures solutions que l'agrégation des solutions des sous-problèmes; ce constat est bien démontré dans la littérature scientifique pour les problèmes de planification des réseaux [11]. Il est utile de noter que ces problèmes de planification, consolidation et migration sont des problèmes NP-difficiles [8] car le problème NP-difficile « bin-packing » [9] peut leur être réduit.

La qualité de service (QdS) est un des aspects importants à considérer lors de la planification, consolidation, et migration des VMs. Cette notion de qualité de service n'est pas unique et elle est dépendante de ses facteurs de mesure par exemple le temps de réponse aux requêtes des usagers, le taux d'erreur dans les données reçues par les usagers, etc. Lors de cette thèse, nous nous intéressons à la qualité totale d'un réseau de VMs; de même, nous nous intéressons à la diminution des violations des contrats de service pour l'ensemble de ces VMs étudiées. L'optimisation du trafic des VMs migrées permet de diminuer les délais de transfert et d'arrêt des VMs pour répondre aux objectifs de l'amélioration de la qualité totale de service de ces VMs. La prise en considération des contrats de service permet aussi d'assurer un minimum de qualité de service auprès des usagers des services virtuels de ces VMs.

Lors de cette thèse, les problèmes de migration simultanée de plusieurs VMs sont considérés comme des problèmes d'optimisation s'intéressant à la diminution des temps d'arrêt des services des VMs, de leurs temps de transfert, des violations des contrats de service, etc. Généralement, ces

problèmes « multi-objectifs » nous donnent un ensemble de bonnes solutions de compromis entre les différents objectifs.

2.2 Répartition des charges (Load Balancing) et les problèmes de gestion dynamique dans les réseaux virtuels

Un des points les plus intéressants des technologies de la virtualisation est « l'approvisionnement dynamique »; autrement dit, les ressources physiques peuvent être regroupées dynamiquement en entités logiques ou virtuelles afin de répondre aux besoins spécifiques de chacune de ses applications; ceci n'est possible qu'à l'aide des techniques de séparation entre les applications et les ressources physiques, et aussi à l'aide des ressources physiques capables d'être approvisionnées dynamiquement [12]. Ainsi, la virtualisation des serveurs et les migrations en temps réel des VMs permettent aux services d'avoir une disponibilité plus élevée (High Availability - HA) et un recouvrement de pannes plus efficace (Disaster Recovery – DR) [13].

Cependant et selon Vijay et al. [7], l'infrastructure actuelle des systèmes informatiques ne permet pas un dynamisme en temps réel qui est nécessaire pour les infonuagiques [14] à cause que : (i) actuellement, l'administration des systèmes se fait manuellement afin de profiter de l'expertise des administrateurs des technologies de l'information (TI) sauf qu'elle génère une « latence humaine » retardant le dynamisme nécessaire pour l'infonuagique; et (ii) la gestion basée sur des règles n'est pas nécessairement totalement automatisable, car ces règles doivent être programmées par des experts en administration de système; ceci complique alors la gestion automatisée de ces règles surtout que le trafic varie rapidement et difficilement prédictible.

De même et selon Vijay et al. [7], les méthodes actuelles d'utilisation des VMs (c.-à-d. de leurs applications, de leurs systèmes d'exploitation, de leurs images de disques de stockage, etc.) ne permettent pas une bonne distribution des ressources partagées, et cela est dû principalement à l'approche centralisée utilisée actuellement (server-centric computing paradigm). Aussi, selon ces auteurs, les applications idéales d'un réseau virtuel doivent se construire comme une collection des services composés, décomposés ou distribués à la volée; ainsi, chaque service est considéré comme un processus individuel faisant partie du flux de cette application; ceci permet à ces services d'être :

(i) optimisés pour répondre efficacement aux exigences de performance et de latence; et (ii) flexibles pour approvisionnement dynamique.

2.3 Contrats de niveaux de service et vérification des niveaux de service (Service Level Checking - SLC)

Selon Chazalet et al. [6], les problèmes et les solutions liés à la vérification du respect des contrats de service sont désignés dans la littérature scientifique par le terme de « la vérification du niveau de service ». Selon ces auteurs, la vérification d'un niveau de service est originaire de la notion du « dépassement de seuil » (Threshold Crossing) [15]. Cette notion de SLC a pris de l'importance avec l'introduction et la définition de la notion des accords sur les niveaux de service [16]. Généralement, un SLC concerne un service cible et un système permettant de collecter, de surveiller des informations, et de vérifier leur conformité aux SLAs. Parmi les produits actuels de SLCs, nous citons les cadres d'applications « Model4SLA » [17], « WSLA » [18], « IBM Tivoli Management Suite » et « HP Service Advisor ».

Selon Chazalet et al., un SLC permet d'envoyer des informations à un administrateur d'un service cible afin de lui permettre de prendre des décisions de reconfiguration, de sélectionner dynamiquement et en temps réel des fournisseurs de service, de lancer des systèmes de boucles autonymiques, et de finaliser des contrats.

Selon Chazalet et al., l'analyse des techniques existantes de la « vérification du niveau de service » montre que les architectures proposées actuellement sont généralement monolithiques; parmi ces architectures monolithiques, nous citons les cadres d'applications: « Model4SLA » de l'Université de Twente [17], « WSLA » de la division « IBM Research » [18], « Layer7Technologies » [19], « WSOM » (Web Service Online Monitoring) de l'Université de Pékin [20]. L'entreprise HP [21] et « Technical University of Catalonia » [22] proposent des architectures en multicouches semblables à l'architecture 3-tiers proposée par Chazalet et al., sauf qu'elles n'utilisent pas la notion de médiation. Selon ces auteurs, la médiation est « un module logiciel exploitant des connaissances encodées à partir d'un ensemble de données dans le but de créer des informations pour des applications s'exécutant sur des couches de niveaux supérieurs » [23]. Cette définition est probablement la première définition du concept de la médiation.

Afin d'intégrer la médiation au concept SLC, Chazalet et al. présentent une architecture à trois couches comportant :

i. **une couche « cœur de surveillance » (Core Monitoring)** collectant et traitant les données des différents services afin d'obtenir des valeurs des indicateurs de bas niveau;
ii. **une couche « de collection de données » (Data Collector)** contenant des chaines de médiation ayant comme entrées des indicateurs de bas niveau collectés par la couche « cœur de surveillance »; ces indicateurs sont traités afin d'obtenir des indicateurs métiers de haut niveau répondant aux objectifs des niveaux de service spécifiés dans les SLAs du service concerné;
iii. **une couche « de surveillance de service » (Service Monitoring)** contenant un ensemble d'applications permettant la surveillance des données avec des préoccupations métiers. Ces applications peuvent être des applications de gestion, des gestionnaires autonomes, des SLCs, des sélectionneurs dynamiques des services, ou des qualificateurs de performance. Cette couche utilise des indicateurs de haut niveau produits par la couche de collection de données. Pour cette raison, une grande partie des implémentations de cette couche comporte des SLCs [6].

Chazalet et al. [6] étudient « les contrats sur les niveaux de service contractés » compatible avec les spécifications « WS-Agreement spécification » [24]. Ainsi, un SLA doit avoir un identifiant, une date de début, une date de fin, et un ensemble d'objectifs concernant le niveau de service (Service Level Objectifs - SLO). Chaque SLO doit avoir son propre identifiant, son information ciblée, son opérateur de comparaison, son seuil pour la valeur de l'information, et sa description.

D'après Chazalet et al. [6], le concept de la médiation des données nécessite une définition stricte des données et de leurs formats, échangés à chaque couche. La couche « collectrice des données » permet des constructions progressives d'indicateurs. La couche de « surveillance des services » contient parfois un système de vérification des conformités aux SLAs. Cette dernière couche de collection de données peut également être utilisée pour collecter et synchroniser/corréler des données provenant de plusieurs ensembles de la couche « cœur de surveillance ».

2.4 Méthodes d'optimisation des problèmes des machines virtuelles

La résolution des problèmes NP-difficiles (comme ceux de la planification, consolidation et migration des VMs) sont résolus généralement par trois types de techniques : (i) techniques de résolution exactes; (ii) techniques de résolution approximatives; ou (iii) techniques hybrides utilisant à la fois des techniques exactes et leur approximation. Ces trois techniques sont détaillées ci-dessous. Ces trois techniques peuvent être testées et comparées en utilisant des simulateurs comme CloudSim [25] (voir la section 2.9).

2.4.1 Méthodes exactes de résolution des modèles mathématiques

Afin d'obtenir des solutions optimales pour des problèmes NP-difficiles, plusieurs techniques exactes existent, mais elles sont plus efficaces pour des ensembles des problèmes avec des petits ensembles. Cependant, ces techniques exactes sont peu efficaces pour des grands ensembles de problèmes puisque leur temps de calcul varie exponentiellement avec leurs tailles [26]. Parmi ces méthodes exactes, nous trouvons des méthodes d'énumération de toutes les solutions possibles; nous recensons aussi des techniques de division intelligente de l'espace de solutions possibles afin de trouver les optimums [27] [28]. Ces solutions exactes répondent à des objectifs à atteindre et des contraintes à respecter [29]. L'ensemble des contraintes à respecter permet de limiter l'ensemble des solutions possibles pour en trouver celle qui maximise ou minimise la fonction objectif respectant des contraintes [30] [31] [32]. Parmi les logiciels connus pour la résolution exacte des problèmes NP-difficiles, nous utilisons CPLEX [33] pour trouver des solutions optimales pour l'ensemble des problèmes cités dans ce document en utilisant la méthode de la programmation en nombres entiers mixte (Mixed Integer Programing – MIP).

2.4.2 Méthodes d'approximation des modèles mathématiques

Les méta-heuristiques sont des techniques utilisées pour optimiser les recherches des optimums sans avoir à réaliser une recherche exhaustive de toutes les solutions possibles. La littérature scientifique est riche de méta-heuristiques et de leurs applications; par exemple, nous pouvons citer les heuristiques de recherche locale et de recherche taboue (TS) [34]. Il existe aussi des heuristiques de colonies de fourmis [35] [36] [37] [38], de recuit simulé [39], de recherche à voisinage variable (VNS) [40], des essaims particulaires [41] [42] et des algorithmes génétiques (GA) [41] [43] [44] [45]. À partir de ces méta-heuristiques, il est possible d'implémenter des heuristiques pour un

problème NP-difficile. Ces heuristiques permettent de trouver des solutions simples et peu consommatrices des ressources, et elles permettent de trouver des solutions très proches des solutions optimales dans des délais acceptables [46] [47]. Ces heuristiques sont dérivées de leur méta-heuristique et ensuite personnalisées en fonction du problème à résoudre. Ces heuristiques définissent des régions à traiter à chaque itération de leur exécution, et elles définissent des types intelligents de déplacements entre une région de solutions et ses voisines.

2.4.3 Méthodes hybrides de résolution des modèles mathématiques

L'intégration des méthodes exactes dans des heuristiques permet de créer de nouvelles techniques hybrides combinant les avantages de chacune d'elles [26] [48] [49]; cette combinaison est possible par exemple en explorant des régions de solutions éventuellement intéressantes à l'aide d'une heuristique; cette exploration est ensuite suivie par une recherche de solution par une méthode exacte [48].

2.4.4 Optimisation multi-objectifs

- **MCDA : Approche distribuée de la gestion dynamique des ressources des VMs**

Yazir et al. [50] proposent une nouvelle approche de la gestion dynamique des ressources. Cette approche est basée sur une architecture distribuée de plusieurs « agents » locaux. Pour valider leurs résultats, ces auteurs utilisent des simulations avec des distributions de données uniformes, normales, exponentielles, de Poisson ou Pareto. Ils simulent les comportements habituels dans les réseaux selon les techniques décrites par Paxson et al. [51]. Les résultats de Yazir et al. montrent, entre autres, que les migrations multiples de VMs sont très coûteuses pour les technologies utilisées actuellement. Il est utile de noter aussi que ces auteurs fournissent des résultats intéressants pour la flexibilité et la modularité des migrations de plusieurs VMs indépendantes; cependant, ces auteurs n'étudient pas le cas des VMs interdépendantes. En résumé, l'approche utilisée par ces auteurs a plusieurs avantages pour les migrations de VMs indépendantes dont :

i. elle est « évolutive », car les calculs, pour la gestion des ressources, se font localement et ils ne sont pas centralisés;

ii. elle est « faisable » dans les centres de données ayant plusieurs machines physiques, car elle minimise le nombre des migrations des VMs nécessaires pour réajuster les ressources;

iii. cette approche est « flexible », car elle permet d'assigner des facteurs pour chacun des paramètres d'évaluation des performances. Ainsi, Yazir et al. utilisent la méthode MCDA basée sur PROMETHEE [52]. Grâce à cette technique MCDA, ces auteurs réussissent à utiliser moins de migration de VMs contrairement aux approches FF (First Fit) et FFD (First Fit Decreasing).

Il est intéressant de noter que l'approche MCDA, présentée par Yazir et al. [50], permet une meilleure redistribution des ressources que celle permise par FF et FFD (voir les figures 6 et 7 de leur article [50]), ce qui réduirait le nombre de violations des contrats SLAs dues à la diminution des machines physiques utilisées [52] [53]. Les figures 8 et 9 de leur article [50] montrent aussi que l'approche MCDA est plus flexible que l'approche de la distribution simple (Simple Distributed — SD) en ce qui concerne l'attribution de poids à chacun des facteurs d'évaluation des performances (par exemple le nombre des migrations requises des VMs, le nombre de PMs requis, etc.). Ainsi, l'approche MCDA de ces auteurs permet de réduire les temps de réponse, l'utilisation des mémoires et bandes passantes, et les violations des SLAs. De même, ces auteurs nous montrent que l'approche MCDA peut être utilisée pour un réseau de grands ensembles de VMs (voir les figures 4 et 5 de leur article [50]; cependant, ces auteurs ne traitent ni des critères propres aux VMs, ni des violations des SLAs, ni des contraintes des VMs interdépendantes, et ils ne traitent que PROMETHEE comme méthode MCDA.

2.5 Techniques actuelles utilisées pour les migrations en temps réel des VMs

2.5.1 « Pré-copie par variation des taux de trafic » et « post-copie » comme techniques de migration en temps réel des VMs

Christopher et al. [10] décrivent deux des techniques de migration de VMs qui sont l'approche conservative de migration et l'approche de migration par variation des taux de trafic. Ces auteurs présentent le concept de « variation des taux de trafic » (writable working set - WWS) permettant de mieux contrôler les moments de transfert des données d'une VM entre deux machines physiques

hôtes; cependant cette technique WWS ne peut pas s'appliquer comme telle pour la migration de machines virtuelles interdépendantes vu que le traitement individuel de chacun des WWS de chaque VM implique plus de problèmes qu'il n'en résout.

Il est important de noter que Christopher et al. [10] se basent sur les travaux existants de NomadBIOS [54] de la technique « pré-copie » tout en lui ajoutant la notion de WWS; ainsi, ces auteurs améliorent l'approche « pré–copie » itérative en dépit de l'augmentation de la quantité nécessaire d'entêtes des paquets de transfert des données. Les principales problématiques de cette approche « pré-copie » itérative, améliorée par WWS, sont les difficultés de reconnaitre les blocs de mémoire à transférer par itérations entre des machines physiques hôtes, et les difficultés de trouver les blocs (qui sont souvent modifiés) à transférer lors de la dernière itération de la migration en temps réel d'une VM.

De même, il est utile de noter que Christopher et al. [10] présentent plusieurs techniques et méthodes de migration comme : (i) la « migration gérée » permettant une migration du contenu de n'importe quelle VM entre n'importe quelles PMs, et (ii) « l'auto-migration » où aucune modification n'est imposée à une VM migrée. De même, ces auteurs constatent que, à l'inverse d'une migration gérée où une VM est suspendue pour avoir un checkpoint consistant, l'auto-migration permet la continuité du fonctionnement d'une VM lors de sa migration. Comme résultat, ces auteurs présentent une amélioration de la migration « pré-copie » en utilisant la technique de variation des taux de transfert des données.

Hines et al. [55] présentent une implémentation et une évaluation de la technique « post-copie » pour les migrations en temps réel des machines virtuelles dans un réseau LAN. Contrairement à la technique « pré-copie », cette nouvelle approche « post-copie » transfert les copies de mémoire après le transfert de l'état du processeur. Cette approche « post-copie » réduit le délai total de la migration d'une VM tout en maintenant son fonctionnement en cours de sa migration. Leurs résultats, basés sur l'utilisation de l'hyperviseur Xen, montrent que cette approche « post-copie » améliore les résultats des migrations des VMs par rapport à l'autre approche « pré-copie » surtout en ce qui concerne le nombre de pages transférées, le délai total de la migration en temps réel d'une VM, et la surcharge du réseau. Ces auteurs utilisent plusieurs stratégies pour minimiser les erreurs de pages à travers un réseau; par exemple, ils ne transfèrent que des copies des pages non-vides.

Ces deux techniques « pré-copie » et « post-copie » permettent de migrer des VMs en temps réel; cependant, elles ne permettent pas les migrations inter-domaines et les migrations de toutes les ressources utilisées par ces VMs migrées; de même, ces deux techniques ne sont pas testées et validées avec les techniques de l'optimisation par para-virtualisation.

2.5.2 WAVNet et « ARPs non sollicités » comme techniques de transfert de données dans un réseau WAN

Zhang et al. [2] proposent un mécanisme appelé « WAVNet » de transfert de données sur des réseaux WANs. Ce mécanisme résout le problème de connexion bidirectionnelle entre des réseaux connectés derrière des « NATs/Firewalls ». Ces auteurs réussissent à effectuer des communications de bonne qualité au sein d'un réseau WAN, grâce à la fluidité des communications (même derrière des « NATs/Firewalls »), la diminution des entêtes dus à la virtualisation, et la conception d'un protocole de découverte des VMs nécessitant des ressources supplémentaires. Ces auteurs décrivent aussi certains outils existants résolvant les problèmes des communications bidirectionnelles dans des réseaux WANs, sauf que ces outils présentés sont spécifiques et ils ne sont pas généralisables. La technique WAVNet [2] utilise une couche au-dessus de l'IPv4 pour contourner les « NATs/Firewalls », et elle permet de connecter deux hôtes d'un réseau WAN comme s'ils sont sur un même réseau LAN par la technique de « UDP hole punching. ». En utilisant cette technique, ces auteurs réussissent à surpasser, par des messages ARPs, les barrières des problèmes de connexions causés par les NATs dans des réseaux WANs. Ces messages ARPs remplacent les techniques de diffusion par DHT qui augmentent les temps d'arrêt des services hébergés sur des VMs migrées. Ces auteurs définissent aussi un modèle mathématique pour déterminer les emplacements des hôtes destinataires des VMs migrées et ils présentent aussi une heuristique afin de construire de bons clusters pour les hôtes destinataires. Ces auteurs démontrent, par différentes expérimentations, que la technique WAVNet est plus performante que la technique IPOP [56] dans des réseaux WANs congestionnés ou non-congestionnés. Cependant, les techniques utilisées par ces auteurs traitent individuellement les migrations des VMs, et elles ne testent pas les migrations simultanées de plusieurs VMs. De même, ces auteurs ne traitent que la migration des applications connectées par MPI [57], et ils n'abordent pas d'autres techniques de communication entre des VMs en cours de migration.

Milojicic et al [58] présentent une vue globale de certains avantages et difficultés de la migration en temps réel des VMs. Ils décrivent, par exemple, les problèmes des dépendances résiduelles sur les hôtes physiques des VMs incluant : (i) des descriptions des fichiers ouverts; (ii) des segments des mémoires partagées, et (iii) d'autres ressources locales. Ces dépendances sont indésirables lors des migrations des VMs, car elles influencent négativement les performances des processus migrées [10] et des disponibilités des machines physiques hôtes.

De même, Christopher et al. [10] présentent les difficultés de copier des disques locaux et des connexions réseaux (avec ou sans redirection [59]) d'une VM migrée entre deux hôtes physiques; généralement, une VM migrée doit copier tous ses états de connections incluant par exemple les connections TCP et PCBs. Pour résoudre cette difficulté, ces auteurs créent des réponses non sollicitées à des ARPs; ces ARPs non sollicités affirment que les adresses IPs des hôtes sources se sont déplacées vers les adresses IPs des nouveaux PMs hôtes.

2.5.3 CloudNet comme technique de migration dans un WAN en utilisant des VPNs

Selon Wood et al. [1], une connexion par la couche 2 (link layer connections), entre deux sites A et B, simplifie la reconfiguration d'un réseau pendant des migrations de VMs puisque cette connexion permet une définition d'un réseau local unique pour les sites A et B. Parmi les différentes techniques possibles de création de telles connexions, nous citons CloudNet qui utilise des points appelés VPLS dans des réseaux VPNs utilisés couramment dans les entreprises.

Selon Wood et al. [1], dans la majorité des cas, les clients sont déjà intégrés aux centres de données de leurs fournisseurs; cependant, si c'est la première fois que le client déplace une VM sur un nouveau site de son fournisseur, un nouveau point de terminaison VPLS doit se créer. Cette création d'un nouveau VPLS implique des changements de configuration (sur les routeurs du centre de données destinataire) facilement automatisables avec des routeurs modernes [60] [61].

Wood et al. [1] indiquent qu'avec CloudNet, un contrôleur VPN peut maintenir un ensemble de règles pour connecter des nœuds. Comme tous les messages de contrôle de routes passent à travers un contrôleur VPN, ce dernier est capable de contrôler la façon de créer des tunnels formant chacun

des VPLS. De cette façon, chaque client a la garantie de disposer de ressources isolées pour son VPLS, ce qui représente une bonne abstraction pour un réseau virtuel privé.

Wood et al. [1] indiquent aussi que, afin de maintenir les connexions réseau pour des migrations de VMs, CloudNet redirige les messages destinés à ses VMs provenant de leurs anciens centres de données vers leurs nouveaux centres de données. Dans des réseaux locaux, cela se réalise par la transmission (non-sollicitée) des messages ARPs vers la machine physique hôte destinataire; ainsi, le contrôleur met à jour sa table de mappage des adresses MAC [10]. En ce qui concerne les réseaux WANs, cette mise à jour du contrôleur ne peut pas être une solution puisque les machines hôtes source et destinataire ne font pas parties du même réseau LAN ayant un même contrôleur. La solution alternative utilisée par CloudNet dans ce cas de WANs est l'utilisation des VPLS pour router ses messages ARPs entre deux centres de données connectés par Internet; comme résultat, les tables de mappage des contrôleurs des deux sites se mettent à jour et les connections, en cours vers des VMs, sont redirigées vers leurs nouveaux sites hôtes.

2.5.4 MPI et PVM comme protocoles pour des communications entre des machines parallèles

Hussain et al. [57] traitent la comparaison entre les protocoles MPI (Message Passing Interface) et PVM (Parallel Virtual Machine) utilisés pour des communications entre des machines parallèles. Selon leur article, même si PVM est utilisable pour des réseaux hétérogènes et inter-domaines, et même si ce même protocole PVM permet des communications dynamiques, l'autre protocole MPI reste le bon outil pour des communications entre des machines parallèles intra-domaine.

Cependant, si leur article [57] met en évidence les avantages des utilisations de MPI et de PVM, il omet de décrire les inconvénients de l'utilisation de ces outils, par exemple, l'augmentation éventuelle des entêtes nécessaires pour des communications entre des VMs en cours de migration; de plus, leur article ne traite pas :

i. de l'utilisation éventuelle de ces protocoles pour des migrations en temps réel des VMs;
ii. de l'utilisation éventuelle des communications entre des machines virtuelles hétérogènes comme entre Xen [62] et OpenVZ [3] [63];

iii. des avantages et des inconvénients de l'utilisation de ces deux protocoles par rapport à d'autres outils de communication entre des machines parallèles comme CPS, Linda, P4, POSYBL, Express et HeNCE.

Même si l'article de Hussain et al. décrit l'utilisation des interfaces de passage des messages (MPI et PVM), il ne traite pas en détail leur traitement des mémoires partagées, dans des systèmes distribués, utilisées par certains outils comme Linda et P4. Enfin, malgré le fait que cet article réalise une comparaison détaillée entre MPI et PVM, il n'est pas utilisable comme tel pour effectuer une critique solide sur la performance de ces outils et leur faisabilité pour des VMs et pour leurs migrations en temps réel dans des réseaux LANs et WANs.

2.5.5 OpenVZ comme technique de migration en temps réel des applications parallèles

Selon Romero et al. [3], une application distribuée parallèle peut terminer son exécution en fournissant des résultats incorrects si un de ses nœuds échoue dans ses calculs. De même, leur article constate que si la taille d'une application et le nombre de ses nœuds de calcul sont grands, le temps moyen de la mise en panne (Mean Time To Failure - MTTF) de cette application est réduit et la probabilité d'avoir des résultats incorrects devient grande.

Pour ces raisons et selon cet article de Romero et al. [3], les approches traditionnelles de la tolérance aux pannes comme la technique des « points de contrôles » (checkpointing) ne réussissent pas à résoudre les pannes de tels grands ensembles d'applications parallèles. Ainsi, afin de trouver une solution pour cette problématique, cet article propose une nouvelle approche, basée sur la migration de VMs, permettant de migrer les processus d'un nœud défaillant vers un autre nœud capable d'exécuter correctement ce processus; ceci permet de réduire le taux de défaut d'une application parallèle. En résumé, cette nouvelle approche utilise l'environnement OpenVZ de virtualisation afin d'effectuer une migration en temps réel d'applications parallèles à multiprocesseurs.

En conclusion, l'étude de cet article de Romero et al. [3] montre que l'utilisation des migrations peut réussir à réduire le nombre des tâches de correction, améliorant ainsi la performance et la fiabilité. De même, cette étude montre qu'il est possible de réaliser des migrations en temps réel

des applications parallèles sans compromettre l'achèvement de leurs calculs parallèles et leur exactitude.

2.5.6 Synthèse des techniques de migration en temps réel des VMs

Comme indiquée par Richmond et al. [64], les deux approches « pré-copie » et « post-copie » permettent la migration en temps réel des VMs; cependant, il n'est pas facile de confirmer qu'une approche est généralement meilleure que l'autre; le choix entre les deux techniques dépend de la quantité des informations à transférer et de l'état de l'adressage sur le réseau de transfert. Le retard de connexion peut influencer négativement la technique « post-copie » plus que la technique « pré-copie ». Concernant, le transfert inter-domaines de VMs, il est primordial de prendre en considération, lors du choix de la technique de migration, le blocage éventuel des redirections d'adressage IP entre les différents domaines. Ainsi, le choix entre les techniques ne peut se faire qu'en prenant en considération l'état de l'adressage du réseau, les quantités de copies des VMs à migrer, la fréquence d'utilisation des services hébergées, etc.

2.6 Exemples de réseaux virtuels et de leurs architectures

2.6.1 Projets nanoHub, myExperiments, SemEUsE et SERVERY comme réseaux de tests d'applications parallèles

Chard et al. [65] présentent des réseaux virtuels permettant de tester des applications virtuelles parallèles (utilisant des VMs). Parmi ces réseaux, ces auteurs citent le réseau « MyExperiment » pour les biologistes et le réseau « nanoHUB » pour la communauté des nanosciences. Selon ces auteurs, le réseau « MyExperiment » fournit un environnement virtuel de recherche où les collaborateurs partagent leurs recherches et exécutent à distance des flux d'applications scientifiques; ces auteurs décrivent aussi le réseau « nanoHUB » permettant à ses usagers de partager leurs données ainsi que d'exécuter en toute transparence des applications avec des ressources distribuées fournies par TeraGrid, etc. Ces deux plateformes, « nanoHub » et « MyExperiment » mettent en évidence différents scénarios possibles de collaboration entre plusieurs applications. Ces applications utilisent entre autres le mécanisme « OpenID » pour identifier leurs usagers.

Selon Chard et al. [65], en plus de ces deux plateformes, il existe plusieurs autres dont le « réseau social virtuel de stockage » pouvant produire les mêmes fonctionnalités et services que « nanoHub » et « MyExperiments ». Ses services concernent le stockage et le partage des données au sein d'une communauté comme des documents universitaires, des flux de travaux scientifiques, des bases de données et d'analyses, etc.

SemEUsE [66] et SERVERY [67] sont deux autres projets coopératifs ouverts utilisés pour tester des réseaux virtuels. Le projet SemEUsE se base sur un bus des services appelé « Petals ESB » de PetalsLink, alors que le projet SERVERY se base sur un serveur d'applications appelé « JOnAS JavaEE » [68]. Ainsi au-dessus de ces deux plateformes, il est possible de créer des applications tierces; par exemple, il est possible de créer un prototype pour visualiser les résultats de certaines SLCs avec : (i) les cadres d'applications open source « OW2 Dream communication » [69] et « Fractal component model » [70] basés sur l'implémentation « Julia » de Java; (ii) le cadre d'applications JMX [71] pour implémenter des interactions et des communications; (iii) le cadre d'applications « Wicket Web » [72] pour implémenter des IHMs ; (iv) l'outils « OW2 JASMINe » [73] permettant de faire fonctionner des applications dans des contextes autonomes (c.-à-d. avec des boucles autonomiques); et (v) les cadres d'applications « CompositeProbes » [74] et « Cilia » [75] spécialisés pour des SLCs concernant des contraintes métiers. Les auteurs des projets SemEUsE [66] et Severety [67] indiquent qu'il est possible de tester plusieurs scénarios et cas d'utilisation par des prototypes implémentés à l'aide de ces deux projets; par exemple, il est possible de simuler et traiter le cas d'utilisation connu sous le nom NRBC (Nuclear Radiological Biological Chemical) correspondant à la gestion de crise impliquant plusieurs acteurs publics (les pompiers, les secours médicaux, etc.); ces acteurs publics peuvent être représentés dans Petals par des services web où un acteur « autorité locale » est vu comme un consommateur faisant appel à ses agents fournisseurs « les pompiers », auxquels il envoie des messages (fonctionnant en mode « push » ou « pull ») par l'intermédiaire d'un bus ESB.

2.6.2 Ordinateur virtuel distribué comme une nouvelle architecture des infonuagiques

Vijay et al. [7] mettent en valeur l'utilité d'un réseau virtuel auprès de ses parties prenantes. Ainsi, leur article indique que d'un côté les usagers finaux s'attendent à une bonne qualité de service

fourni par les technologies des infonuagiques et à des délais acceptables de réception de ces services; d'un autre côté, les développeurs des services virtuels s'attendent à avoir de bons moyens pour gérer en temps réel leurs requêtes de modification des capacités des ressources mises à leur disposition par leurs fournisseurs des services virtuels; ces fournisseurs sont alors contraints à effectuer simultanément une réductions de leurs coûts totaux et une amélioration de l'agilité des services de leurs centres de données.

Selon ces auteurs [7], les architectures actuelles des infonuagiques ne sont pas évolutives et ne répondent pas nécessairement aux besoins d'un très grand nombre d'usagers. Ceci est dû principalement à l'historique de la mise en place de l'infrastructure actuelle des infonuagiques qui hérite de l'approche centralisée autour d'un serveur central. Comme son évolution s'est réalisée par des améliorations incrémentales de ses plusieurs couches d'automatisation de la gestion des ressources, cette approche centralisée nécessite énormément de connaissances et de travaux intensifs afin de comprendre et d'utiliser efficacement ces couches; par conséquent, ces couches sont devenues inefficaces pour répondre aux demandes actuelles des clients qui changent continuellement de comportements.

De même, selon Vijay et al. [7], le développement actuel, du « matériel d'assistance à la virtualisation » et des techniques de « séparation entre les infrastructures et les applications », permet une meilleure gestion des services virtuels couplés à une meilleure visibilité. Ainsi et grâce à ces techniques, il est possible de concevoir une nouvelle approche axée sur les réseaux (network-centric approach) plutôt que d'avoir une approche centralisée axée sur des serveurs (server-centric approach). Cette nouvelle approche permet de répondre en temps réel aux requêtes des usagers finaux et fournir des services dynamiques, modulables, sûrs, et sécuritaires.

Actuellement, selon Vijay et al. [7], il existe plusieurs difficultés pour profiter du plein potentiel des « approvisionnements dynamiques » des infonuagiques. Ces difficultés sont dues à plusieurs éléments dont : (i) les systèmes d'exploitation actuels basés sur « l'approche centralisée sur un serveur » ne sont pas conçus initialement pour gérer des ressources distribuées, des systèmes de « répartition de charges » (load balancing) et de « clustering », alors que ces approches des infonuagiques devraient normalement se focaliser sur l'optimisation du partage de leurs ressources distribuées; (ii) même si les hyperviseurs actuels permettent à plusieurs VMs d'être hébergées sur

une seule machine physique [76] [77], ces hyperviseurs ne permettent pas la séparation entre la gestion des applications et celle des ressources physiques puisqu'ils ne permettent pas actuellement de gérer des options des ressources des applications (comme leur priorité, leur profil d'utilisation des ressources, etc.); et (iii) les techniques actuelles de virtualisation d'un serveur ne permettent pas le partage des ressources distribuées et elles ne permettent pas de construire un serveur à partir des ressources localisées dans des emplacements physiques différents.

Selon ces auteurs Vijay et al. [7], même si actuellement nous pouvons migrer des applications d'un serveur physique à un autre, cela ne signifie pas nécessairement que nous pouvons partager leurs ressources physiques; pour cette raison, leur article discute de la nécessite d'avoir un vrai « ordinateur virtuel distribué » permettant de partager des ressources distribuées n'importe où elles sont en ne prenant en considération que des contraintes de latence des applications et des services consommés.

En résumé et selon cet article de Vijay et al. [7], la notion d'un « ordinateur virtuel » doit correspondre idéalement à un « pool » de ressources physiques potentiellement distribuées géographiquement sur un ou plusieurs serveurs; ceci permet d'avoir une entité logique dynamique appelée « ordinateur virtuel distribué ».

2.6.3 Nouvelle architecture pour une bonne séparation de la gestion des ressources virtuelles de celle des ressources physiques

Vijay et al. [7] proposent un nouveau modèle de référence d'architecture pour les infonuagiques afin de bien séparer la gestion de ressources virtuelles de la gestion des ressources physiques; cette nouvelle architecture utilise des « médiateurs en temps réel » entre les applications et les ressources. Cette architecture proposée est composée de plusieurs structures dont :

i. **la structure de l'infrastructure de service** composée des deux couches suivantes permettant de fournir équitablement des ressources aux applications virtuelles :
   a. **la couche de la médiation des services distribués** qui est une couche abstraite permettant une autogestion individualisée et autonome pour chacune des ressources distribuées géographiquement;

b. **la couche de la médiation des ressources virtuelles** qui permet de construire des serveurs virtuels logiques avec certains niveaux de garantie de qualité de service par exemple le nombre des CPUs, la quantité des mémoires, la quantité des débits, la durée des latences, la quantité des opérations I/O par seconde (IOPS), la capacité de stockage, etc.

ii. **la plateforme de l'assurance des services distribués** permettant de créer des serveurs virtuels respectant les contraintes de gestion des réseaux (FCAPS) et d'héberger et lancer des systèmes d'exploitation (OS) exécutant des applications virtuelles. Puisque ces serveurs virtuels logiques supportent les contraintes FCAPS, ils assurent des services de médiation automatisés aux applications pour une gestion efficace des erreurs et de la fiabilité (HA/DR), et pour une sécurité et des performances optimisées. Ainsi, les fournisseurs de ces serveurs virtuels peuvent fournir des APIs de gestion (management dial-tone) aux développeurs afin de créer des services (auto-configurables, autocorrectifs et auto-optimisés) composant des flux fonctionnels autogérés et indépendants de l'infrastructure physique;

iii. **la plateforme de livraison des services distribués** comportant essentiellement un moteur de flux orchestrant les composantes distribuées des flux applicatifs (business workflows);

iv. **la plateforme de création des services distribués** fournissant des librairies nécessaires pour développer et créer des applications. Ces applications pourraient être des collections de plusieurs services composées, décomposées et distribuées à la volée à plusieurs serveurs virtuels. Ces derniers sont créés et gérés automatiquement par la plateforme de l'assurance des services distribués;

v. **la couche de médiation de l'intégration d'un système existant** qui permettrait d'intégrer et de supporter des applications existantes.

Vijay et al. [7] réussissent à démontrer que leur modèle, proposé comme une architecture infonuagique, est faisable et intéressant surtout pour : (i) l'approvisionnement des ressources selon les profils des applications; et (ii) la médiation des services dynamiques respectant des contraintes FCAPS. Cependant, cette démonstration s'est faite avec un prototype logiciel et non pas par une implémentation réelle, et elle est testée seulement dans un réseau local (LAN).

2.6.4 Réseaux virtuels sociaux comme une architecture pour des réseaux virtuels

Notre étude traite la migration des VMs parallèles dans des réseaux LANs/WANs. Elle pourrait aussi tester ses résultats dans le cas d'un réseau virtuel social afin de bénéficier de ses avantages en ce qui concerne la sécurité. Pour cette raison, nous nous intéressons à la notion des infonuagiques sociaux traitée par Chard et al. [65]. Ces auteurs présentent dans leur article les réseaux sociaux comme des connexions entre leurs usagers afin de partager leurs informations et de créer des organisations virtuelles dynamiques. Leur article est une des études rares traitant un réseau social pour effectuer une gestion dynamique de ses usagers, de leur authentification, et de leur utilisation du système.

Comme indiqué par Amazon [78], les membres d'une communauté virtuelle peuvent construire ensemble des applications virtuelles en les déployant sur une infrastructure d'un fournisseur tiers d'infonuagiques (comme Amazon Web Services). Aussi, il est possible d'exposer des applications virtuelles aux communautés virtuelles par l'outil ASPEN (Automated Service Provisioning Environment) [79] et les technologies Web 2.0.

Chard et al. [65] proposent un réseau virtuel social dynamique se basant sur la confiance préétablie et formée à travers des relations d'amitié dans un réseau social. Ainsi, des amis dans un réseau virtuel social peuvent partager leurs ressources. Ces auteurs présentent, implémentent, et testent plus particulièrement leur propre réseau virtuel social de stockage (avec le respect des contrats SLAs). Ce réseau virtuel facilite les communications et les partages des ressources de stockage entre un nombre important de membres d'une communauté d'amis virtuels en se basant sur leur réputation dans ce réseau social.

Ainsi et selon Chard et al. [65], les usagers d'un réseau virtuel social peuvent avoir des crédits leur permettant de gagner des ressources virtuelles (comme c'est déjà le cas avec PlanetLab [80]). Cette implémentation est réalisée sous la forme d'une application Facebook permettant de profiter de l'infrastructure de la plateforme Facebook, de ses APIs, de ses mécanismes d'identification, de ses stratégies d'accès, de ses outils de développement (c.-à-d. les fonctionnalités de son interface RESTlike, le langage FBML (Facebook Markup Language), le FBJS (Facebook JavaScript), etc.).

Comme les autres applications de stockage de données, cette application gère ses risques concernant la sécurité des données en utilisant des mécanismes existants comme le « sandboxing » et la réplication.

Au niveau de son implémentation et comme indiqué par Chard et al. [65], cette application de réseau social virtuel utilise des services web en se basant sur le WSRF (Web Service Resource Framework) exécuté sur « Globus WS-core/Tomcat ». D'autres outils ont été utilisés pour implémenter cette application comme le service de stockage gRAVI (Grid Remote Application Virtualization Interface) [81], SORMA [82] pour la création de WSAgreements avec des outils EJSDL [83] [84], le cadre d'applications des enchères DRIVE (Distributed Resource Infrastructure for a Virtual Economy) [85] utilisant le protocole « reverse Vickrey », des pages JSPs avec JSON, un serveur « Tomcat 6 » pour héberger des pages JSP, « Mozilla Firebug » pour mesurer les latences d'ouverture des pages web, des appels asynchrones directs par Ajax, etc.

Chard et al. [65] montrent que leur application, du réseau virtuel social de stockage, est capable d'exécuter, avec une mémoire de 1 GB, plus de 2000 entrées en moins de deux secondes; cependant, les performances de cette application virtuelle peuvent être affectées par l'augmentation du nombre d'enchères simultanées réalisées par un agent des enchères (Auction Manager – AM); cet inconvénient pourrait être résolu par l'augmentation du nombre d'agents AMs afin de réduire le nombre d'enchères effectuées par chacun d'eux.

2.7 Migrations des benchmarks HPL et OMEN comme exemples de migration de machines virtuelles

Romero et al. [3] présentent les résultats de leurs expériences concernant des migration de deux systèmes parallèles HPL (High Performance Linpack) [86] et OMEN [87] [88]. Suite à ces résultats, nous remarquons que la migration simultanée, de plusieurs VMs exécutant des parties de l'application HPL, provoque une grande augmentation du temps d'exécution de l'application allant jusqu'à 54 fois. Pour cette raison, à notre avis, il est utile de trouver des mécanismes permettant de minimiser cette augmentation des temps d'exécution des applications qui ont une fréquence très élevée d'interaction entre leurs VMs. Selon notre constat, nous remarquons que :

- les auteurs Romero et al. ont étudié des applications utilisant les protocoles MPI et MPICH2 [57]; cependant, ils n'ont pas traité ni des applications utilisant le protocole PVC, ni des applications inter-domaines, ni des applications avec des communications indirectes;
- les auteurs Romero et al. ne donnent pas de pistes possibles pour l'amélioration des délais d'exécution des services des VMs parallèles migrées en temps réel. Leur article pourrait traiter par exemple l'amélioration de la tolérance aux pannes de l'outil MPI; pour cette raison, nous pensons qu'il existe des opportunités d'amélioration de ces délais en améliorant l'outil MPI, voire aussi l'outil PVM [57], afin d'obtenir des migrations performantes en temps réel des VMs parallèles et interdépendantes dans des réseaux LANs et WANs.

2.8 Difficultés de la gestion des composantes d'un réseau virtuel

2.8.1 Stockage virtuel en réseau

Selon Gartner [89], la virtualisation du stockage réseau est plus ancienne que les serveurs virtuels; ainsi, grâce à l'utilisation du protocole FC (Fibre Channel) et du protocole SAN (Fibre Channel-based Storage Area Networks), plusieurs solutions spécialisées en virtualisation permettent des connectivités rapides pour stocker des données et les répliquer. De même, des techniques optimisées existent pour une disponibilité accrue et un recouvrement efficace des pannes (HA/DR); cependant, ces outils de stockage réseau augmentent les coûts et les complexités de sa gestion (surtout les tâches d'un administrateur SAN). Cette situation freine actuellement le déploiement massif des techniques de virtualisation de stockage des données.

2.8.2 Facteurs de complexité de la gestion des composantes de la virtualisation

Selon Vijay et al. [7], la complexité de la gestion des composantes de la virtualisation est due à plusieurs facteurs dont : (i) le grand nombre des nouvelles VMs (Virtual Machine Sprawl) pouvant être créées facilement sur des serveurs virtuels; (ii) le grand nombre d'applications pouvant être hébergées dû à la faiblesse des coûts d'installation de leurs VMs hôtes; ces nombreuses applications peuvent être des répartiteurs de charges (load balancers), des routeurs, etc. (iii) la difficulté de gérer

simultanément du trafic inter-VMs et du trafic classique inter-machines physiques; et (iv) l'hétérogénéité des environnements des systèmes d'exploitation (OS) installés sur des VMs.

Comme résultat, Vijay et al. constatent que la complexité accrue de la gestion des systèmes virtuels actuels peut diminuer les bénéfices de la consolidation par des techniques de virtualisation. Finalement et selon ces auteurs, l'apparition du matériel d'assistance à la virtualisation (HAV) est un pas en avant permettant aux hyperviseurs de gérer mieux les accès équitables aux ressources physiques locales, et aussi à celles hébergées sur d'autres serveurs distants et connectées en réseaux.

2.8.3 Utilisation non optimale des capacités des machines physiques

Les auteurs Shivani et al. [90] traitent les migrations en temps réel des états d'une VM migrée entre deux machines physiques. Ces auteurs traitent aussi le cas où les ressources (puissances de calcul, mémoires, etc.), utilisées par une VM migrée, changent lors de sa migration.

De plus, ces auteurs montrent que les solutions actuelles, de migration en temps réel des VMs, n'utilisent pas efficacement les ressources des machines physiques; de même, leur article montre que généralement une VM migrée n'utilise que la base minimale commune des ressources des machines physiques hôtes sources et destinataires. Enfin, nous constatons que ces auteurs posent beaucoup de questions de recherche, mais ils ne leur fournissent pas des réponses claires et prouvées.

2.9 Simulation des machines virtuelles comme technique de test des optimisations des migrations des machines virtuelles

Afin de tester les comportements des machines virtuelles suite à l'application des techniques d'optimisation, plusieurs études [91] [92] [93] utilisent des simulateurs qui permettent de déduire les résultats de ces techniques et de déterminer leurs limitations et en déduire les améliorations possibles. Lors de ces simulations, les paramètres utilisées sont issues des comportements réels des machines virtuelles déjà installées dans des centres de données [93]; les types de serveurs simulés sont différents afin de reproduire les divers serveurs du marché des infonuagiques; les processeurs simulés sont sélectionnés à partir d'un ensemble de processeurs Intel (par exemple Atom, i5, i7 et

Xeon) [94] avec un nombre différent de cœurs, de cache, et de consommation d'énergie [93]. Lors de nos tests, nous utilisons un simulateur ressemblant à celui de Goudarzi et al. [93] afin déduire les résultats de nos modèles mathématiques et les comparer aux techniques existantes dans la littérature scientifique. D'autres simulateurs existent aussi dans la littérature scientifique comme CloudSim [25].

2.10 Analyse sommaire de la revue de littérature de la problématique de migration en temps réel des VMs interdépendantes

Comme nous constatons dans les sections précédentes, il existe plusieurs outils pour assister une personne à migrer une VM sur un réseau LAN/WAN; ces outils permettent de diminuer les délais de migration en temps réel d'une VM entre deux hôtes physiques. Cependant, les études actuelles n'ont pas traité l'optimisation de la migration groupée en temps réel de plusieurs VMs interdépendantes et parallèles. Les articles de la littérature scientifique traitent généralement la migration en temps réel de plusieurs VMs en faisant abstraction de leurs interactions, ce qui leur simplifie la modélisation au détriment de la qualité de leurs solutions; ceci nous laisse avec des solutions non optimales pour les migrations de VMs interdépendantes, même si elles sont optimisées pour des migrations de VMs indépendantes. Ainsi, l'optimisation de ces migrations en temps réel de plusieurs VMs reste un domaine à étudier; même si certaines études mentionnent bien qu'il est possible de migrer plusieurs VMs ensemble [3] [4], elles ne précisent pas s'ils réussissent à le faire dans des délais acceptables avec le respect des SLAs surtout dans le cas des VMs interdépendantes.

La littérature scientifique actuelle est limitée en ce qui concerne la planification, la consolidation et la migration en temps réel de plusieurs VMs. Les chercheurs scientifiques traitent une seule VM à la fois lors du traitement de sa planification et sa migration en temps réel. Les problèmes de planification, consolidation et migration en temps réel de plusieurs VMs sont peu traités selon notre revue de littérature. De plus, les modèles proposés ne traitent pas toutes les caractéristiques du problème par exemple les interdépendances entre des VMs migrées simultanément, les violations des SLAs et la qualité globale des services virtuels d'un réseau de VMs.

Dans la littérature scientifique étudiée, les auteurs sont plutôt intéressés par la planification et la migration en temps réel d'une VM. En effet, les machines virtuelles étant des technologies relativement nouvelles, les techniques de leur utilisation efficace, nécessitent de nouvelles innovations. Comme les machines virtuelles ont généralement les mêmes propriétés globales et nécessitent les mêmes besoins en technologies informatiques, il est inspirant de leur trouver une formulation technique globale.

Malgré que les étapes de la migration simultanée en temps réel de plusieurs VMs soient similaires à celles de la migration d'une seule VM (traitée majoritairement par les études scientifiques), ces étapes sont plus complexes à réaliser adéquatement et dans des délais raisonnables. En effet, les VMs, à migrer simultanément, peuvent être interdépendantes et parallèles, et donc elles peuvent avoir des fonctionnalités parallèles et simultanées à offrir à leurs clients; par conséquent, la non-coordination éventuelle entre ces VMs, lors de leur migration simultanée, peut mettre en péril les fonctionnalités des services de l'ensemble de ces VMs. De plus, les VMs, une fois migrées, doivent aussi être capables de continuer à fonctionner ensemble correctement afin de fournir au moins les mêmes qualités de service qu'avant leur migration.

La littérature scientifique comporte peu d'études sur les planifications, consolidations et migrations en temps réel simultanées de plusieurs VMs; ces travaux concernent principalement des VMs indépendantes. Malgré qu'il soit faisable d'appliquer les modèles suggérés dans la littérature scientifiques [50] [91] [92] [93] pour l'optimisation des migrations simultanées de plusieurs VMs, ces modèles ne traitent pas les contraintes d'interdépendance entre ces VMs, le respect des SLAs, et la qualité globale des services virtuels.

Lors de cette thèse, nous intégrons plusieurs objectifs définis précédemment dans le but de définir un modèle global englobant l'interdépendance entre des VMs, les respects des SLAs et la qualité globale de service d'un réseau de VMs interdépendantes. Ce modèle est beaucoup plus difficile à définir et à résoudre qu'une planification, ou migration d'une seule VM; ainsi, notre modèle et ses heuristiques d'approximation sont plus efficaces grâce à la prise en considération simultanément de plusieurs objectifs et de plusieurs contraintes d'interdépendance entre des VMs distribuées. Notre modèle proposé dans cette thèse considère l'optimisation de plusieurs objectifs anti-corrélés comme la diminution des temps d'arrêt, des services virtuels des VMs en cours de migration en

temps réel, la maintenance de la qualité globale de ces services lors de ces migrations, et la minimisation de la pénalité globale aux contrats SLAs.

Donc, lors de cette thèse, nous étudions un cadre global, de la planification, la consolidation, et la migration en temps réel des VMs, utilisable pour plusieurs VMs parallèles interdépendantes migrées simultanément. Ce cadre global intègre plusieurs sous-problèmes comme la garantie de la qualité globale d'un réseau de VMs, le respect des contrats de niveaux de service, et la minimisation des durées de transfert et d'arrêt des VMs migrées en temps réel. Même si la démarche de ce cadre global est plus complexe que celles des études scientifiques actuelles des migrations en temps réel des VMs, elle a l'avantage de minimiser le temps de transfert des VMs parallèles et interdépendantes en préservant les interactions entre ces différentes VMs migrées simultanément.

Enfin, il est utile de noter que le but de notre thèse est de définir des modèles et heuristiques pratiques d'optimisation des problèmes de planification, consolidation et migration en temps réel des VMs interdépendantes.

2.11 Conclusion

Dans ce chapitre, nous avons décrit l'état de l'art des réseaux infonuagiques, des machines virtuelles, de leur planification, consolidation, et migration en temps réel afin de démontrer l'importance de leurs problématiques.

Le chapitre suivant décrit notre démarche pour la résolution de notre problématique de la planification, consolidation, et migration en temps réel des machines virtuelles interdépendantes.

CHAPITRE 3 DÉMARCHES DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE ET ORGANISATION GÉNÉRALE DU DOCUMENT INDIQUANT LA COHÉRENCE DES ARTICLES PAR RAPPORT AUX OBJECTIFS DE LA RECHERCHE

Après avoir exposé les objectifs de notre recherche lors du premier chapitre de l'introduction, nous détaillons dans ce chapitre la démarche qui nous permet d'y arriver.

Le but principal de cette thèse est de développer un cadre global de placement d'un ensemble de VMs interdépendantes. Ce placement de VMs s'effectue au cours de leur première installation et postérieurement. Notre étude a donc trois objectifs suivants : (i) la résolution de ce problème NP-difficile de placement de VMs lors de leur première installation; (ii) la résolution du problème de maintien d'une bonne qualité globale de l'ensemble de VMs lors de leur replacement à l'aide de migration en temps réel; et (iii) enfin, modéliser et tester la consolidation des services virtuels hébergés par des VMs en utilisant, comme base de tests, une configuration existante (décrite dans des fichiers des traces de Google) des services et des machines physiques.

3.1 Premier objectif de recherche

Le premier objectif de notre recherche est la conception d'un modèle mathématique et de son heuristique d'approximation pour optimiser les planifications d'un réseau de VMs interdépendantes tout en prenant en considération leurs contraintes d'interdépendance. Cet objectif est réussi dans l'article du quatrième chapitre intitulé « On the Design of Large-Sized Cloud Networks Optimized for Live Migration Using Tabu Search Algorithms ». Dans cet article, nous proposons une heuristique de recherche taboue permettant d'approximer notre modèle mathématique de planification de VMs intégrant leurs contraintes d'interdépendance. Notre modèle mathématique proposé reprend l'ensemble du problème de planification d'un réseau de VMs sans avoir à le décomposer en plusieurs sous-problèmes. Ce modèle a comme fonction objectif de minimiser les coûts d'un réseau de VMs tout en respectant les contraintes d'interdépendance entre ces VMs. Lors de l'annexe B, nous présentons plus de tests sur ce modèle mathématique (voir l'annexe A pour son développement détaillé) et son heuristique d'approximation afin de montrer leur efficacité avec plus de scenarios ayant des milliers de VMs.

- **Utilisation des heuristiques d'approximation**

Ainsi, nous avons utilisé une heuristique de recherche taboue pour résoudre le premier objectif de notre étude pour des grands ensembles de VMs (voir le chapitre suivant). Cette heuristique permet d'approximer efficacement notre modèle mathématique de la planification des VMs interdépendantes dans le contexte des très grands ensembles. Cet objectif est réalisé par la démarche détaillée dans le quatrième chapitre présentant l'article de conférence publié et intitulé « On the Design of Large-Sized Cloud Networks Optimized for Live Migration Using Tabu Search Algorithms ». Cette heuristique proposée, comme réponse au premier objectif de recherche, est basée sur la technique de recherche taboue, et elle permet une approximation de notre modèle mathématique par des explorations de plusieurs « régions locales de l'espace de recherche » des solutions possibles. Nous intégrons à cette heuristique, de recherche taboue, des techniques d'aspiration afin de ne pas stagner notre heuristique avec une solution optimale locale et afin d'explorer efficacement notre espace de recherche de solutions possibles.

Afin de tester les résultats de notre premier objectif de recherche, de planification optimale des VMs interdépendantes résolu dans l'article de conférence publié et repris dans le chapitre suivant, nous avons comparé les solutions de notre heuristique de recherche taboue avec celles de la méthode exacte pour conclure que notre méthode est plus efficace pour des grands ensembles de tests où le solveur exact CPLEX [33] n'arrive pas à trouver rapidement de bonnes solutions dans des délais acceptables avec la méthode de programmation en nombres entiers mixte (MIP). Ceci est dû principalement à l'augmentation exponentielle de l'espace de recherche des solutions de ce problème NP-difficile de placement de VMs car le problème NP-difficile « bin-packing » [9] peut être réduit à ce problème. Il est utile de noter que CPLEX est plus efficace pour des petits ensembles de tests où ce solveur trouve des solutions optimales dans des délais raisonnables. L'annexe B valide les résultats de notre heuristique avec des ensembles de tests plus grands que ceux du quatrième chapitre.

3.2 Deuxième objectif de recherche

Le deuxième objectif de notre recherche est la conception d'un modèle mathématique « multi-objectifs », de sa relaxation en un modèle « mono-objectif », et de son heuristique d'approximation utilisables pour une amélioration de la qualité globale de service d'un réseau de VMs tout en diminuant les temps de migration des VMs interdépendantes. Cet objectif de modélisation

mathématique est réussi dans l'article de conférence publié et repris dans le cinquième chapitre intitulé « Real-Time QoS Issues in Live Migration of Interdependent Virtual Machines ». Dans cet article, nous proposons un modèle mathématique « multi-objectifs », relaxé à un modèle « mono-objectif » par la somme pondérée de chaque objectif, maximisant le total des qualités de service des VMs d'un centre de données virtuel et minimisant les temps d'arrêt des services des VMs migrées en temps réel, de leur délais de transfert, de leurs pénalités sur les non-respects des contrats de niveaux de service. Notre modèle mathématique proposé reprend l'ensemble du problème des migrations en temps réel des VMs interdépendantes d'un réseau de VMs sans avoir à le décomposer en plusieurs sous-problèmes. Ce modèle est ensuite testé, validé et comparé à d'autres modèles existants dans la littérature scientifique.

Concernant ce deuxième objectif de recherche et afin de valider la prise en considération de l'interdépendance des VMs dans nos solutions des migrations en temps réel des VMs interdépendantes, nous les avons comparées avec d'autres solutions publiées dans la littérature scientifique qui ne traitent que des VMs non interdépendantes; nous avons généré des interdépendances entre les services de ces VMs en utilisant la distribution Bernoulli. Comme attendu, nous avons constaté que les solutions considérant l'interdépendance des VMs coûtent moins en dégradation des contrats de niveau de service et de la qualité totale des services virtuels fournies par ces VMs. L'annexe C présente des tests avec des grands ensembles afin de valider les conclusions de la pertinence des résultats du cinquième chapitre.

3.3 Troisième objectif de recherche

Notre troisième objectif de recherche est la conception d'un modèle mathématique utilisable pour une consolidation des VMs interdépendantes en vue de minimiser la pénalité globale sur les SLAs et de maximiser le profit net global des services virtuels de ces VMs. Cet objectif est réussi dans l'article soumis du sixième chapitre intitulé « Live Placement of Interdependent Virtual Machines to Optimize Cloud Service Profits and Penalties on SLAs ». Dans cet article soumis, nous proposons un modèle mathématique qui a comme objectif de maximiser le profit net total en minimisant la pénalité globale aux contrats de niveaux de service de l'ensemble des VMs et en respectant leurs contraintes d'interdépendance lors de leurs migrations en temps réel. Ce modèle mathématique et son heuristique d'approximation consolident les VMs tout en respectant les contraintes de capacités de leurs machines physiques hôtes en ce qui concerne l'utilisation des

ressources comme les mémoires, les CPUs, les bandes passantes, etc. Ce modèle mathématique est testé avec le logiciel CPLEX [33] alors que notre heuristique est testée avec des simulations d'un réseau de VMs. Afin de valider notre heuristique, nous l'avons comparé avec la configuration d'un ensemble de services déployés et décrits dans des traces de Google [95]. Comme attendu, nous avons constaté que nos solutions sont de meilleure qualité que nos références de comparaison.

3.4 Utilisation des modèles mathématiques

Les modèles mathématiques présentés dans nos différents articles (voir les chapitres suivants) sont testés en utilisant la méthode de programmation en nombres entiers mixte (MIP) du solveur mathématique CPLEX [33] permettant de résoudre d'une manière exacte des problèmes mathématiques NP-difficiles pour des petits ensembles. Les paramètres utilisés pour ce solveur CPLEX et les paramètres des heuristiques sont décrits dans les annexes B et C.

Nos modèles mathématiques proposés pour nos trois objectifs de recherche (détaillés dans le premier chapitre de l'introduction) ne peuvent être résolus avec exactitude que pour des petits ensembles, car le temps de leur résolution augmente exponentiellement avec la taille des ensembles de tests. Pour ces raisons, les techniques d'approximation (comme des heuristiques de recherche taboue) sont très utiles pour trouver de bonnes solutions dans des délais acceptables pour des grands ensembles de VMs.

3.5 Conclusion

Lors de ce chapitre, nous avons détaillé notre méthodologie nous permettant d'obtenir de bons résultats pour nos trois objectifs. Les trois chapitres suivants montrent que les objectifs de cette thèse sont atteints grâce à nos résultats basés sur la méthodologie décrite dans le présent chapitre et qui sont intégrés dans les trois différents articles scientifiques détaillés dans les prochains chapitres. En plus de ces trois chapitres, nous avons exécuté plus de tests et nous avons décrit leurs résultats dans les annexes; ainsi, l'annexe B présente des tests supplémentaires pour le premier objectif basé sur le modèle mathématique (du quatrième chapitre) détaillé dans l'annexe A; de même, l'annexe C présente des résultats supplémentaires pour notre deuxième objectif présenté dans le cinquième chapitre.

Notre travail permet une meilleure appréhension de la planification, la consolidation et la migration des VMs interdépendantes en prenant en considération les contrats des niveaux de service et la

qualité de service fourni par ces VMs. Ainsi, les centres de données virtuels peuvent diminuer les coûts de leurs pannes dues aux non-respects des contraintes d'interdépendance des VMs et des contrats des services. De plus, il est intéressant de noter que de cette thèse présente aussi des avantages économiques, car ses résultats pourront être utilisés pour réduire les coûts de l'utilisation des VMs dédiées au grand public.

CHAPITRE 4 ARTICLE 1: ON THE DESIGN OF LARGE-SIZED CLOUD NETWORKS OPTIMIZED FOR LIVE MIGRATION USING TABU SEARCH ALGORITHMS

Auteurs : Salah-Eddine Benbrahim, Alejandro Quintero and Martine Bellaiche

IEEE - 2014 8th Asia Modelling Symposium (AMS)

Abstract

This work, based on the paper [96], presents and develops a tabu search (TS) algorithm for the planning of interdependent virtual machines (VMs) with regards to data exchange deadline concern between their interdependent hosted applications. The main contribution of this research is the use of a tabu search (TS) heuristic to find near-optimal node positions in a virtual network having interdependent VMs and shared resources. This TS is an interesting solution regarding its precision and feasibility for large-sized virtual networks. The results of the TS proposed in this paper show that it is possible to reduce the planning costs of a virtual network while respecting interdependency constraints of its shared and distributed resources.

4.1 Introduction

Virtual networks, as other network types, offer various services to their end users such as streaming, data and voice applications. These different services are generally hosted by virtual machines (VMs) loosely coupled to physical machines (PMs). This service virtualization has a lot of advantages for data centers such as Server Consolidation, Centralized and Policy-driven Management of a Heterogeneous Data center Environment, and High Availability.

Optimized virtual networks are a necessary to data centers to provide good quality services. Among the main criteria of optimized virtual networks, efficient communications between their interdependent applications are vital to provide efficient data center services. Interdependent applications of a data center are hosted by interdependent VMs which share between them some of their online and offline data. These online data are shared, transmitted and propagated with a prescribed deadline for each of the interdependent applications.

As other network types, a virtual network can be divided into an access network and a core network (see Figure 4.1). In the access network, VMs (Virtual Machines) [97], VMMs (Virtual Machine

Managers) [98], and SDSRVs (Shared Data and virtual hard disk Servers) host network shared data and virtual hard disks. The core network is composed of MRs (Memory Routers) and HRs (Hard disk Routers). An example of a virtual network, where a user can access an application hosted by a VM, is described in Figure 4.1 where VMMs are intermediate nodes between VMs and SDSRVs and where SDSRVs are connected both to MRs and HRs.

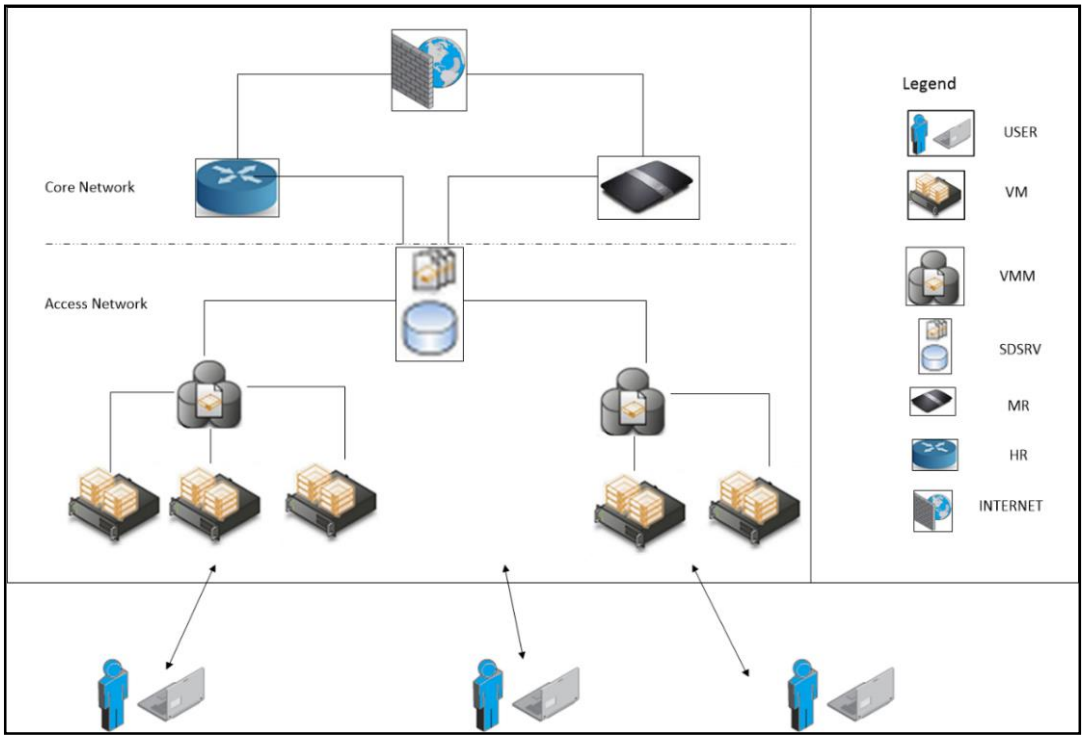

Figure 4.1: Example of interdependent virtual machines' network architecture.

Access networks communicate with core networks via high bandwidth connections of two types. First type connections transfer VMs' online data, while second type connections transfer network shared and offline data. Core networks, composed of memory routers (MRs) and hard disk data routers (HRs), route VMs online data by MRs, whereas HRs route shared and offline data. These MRs and HRs connect their virtual network to Public Data Networks (PDNs) (Figure 4.1).

Core and access networks have a total installation cost composed of: (1) cost of links joining VMs to VMMs, VMMs to SDSRVs, SDSRVs to MRs and HRs; and (2) cost of installed nodes (VMs, VMMs, SDSRVs, MRs and HRs). The problem of global planning of virtual networks (GPV) consists essentially to find a configuration which minimizes the total network installation costs.

This problem is NP-hard and it can be explored by exhaustive search methods that would entail a combinatorial explosion and, therefore, an exponential augmentation of execution times [99].

This paper proposes a tabu search heuristic to efficiently solve this problem of assigning interdependent VMs in virtual networks. Section 4.2 provides background information and related work. Section 4.3 describes our tabu search heuristic. Section 4.4 deals with the experimental results. Finally, Section 4.5 discusses the results while Section 4.6 concludes this study with some possible further work.

4.2 Overview of Cloud Networks and Virtual Machine Allocation Techniques

Among known cloud computing products, there are IBM Blue Cloud [100] and Amazon EC2. Cloud suppliers provide web access to their services to end users thanks to high-speed Internet connectivity. Cloud services can be Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [101]. As cloud infrastructure are flexible, scalable and easy to maintain, their use is continuously increasing. As a result, scheduling and performance management of cloud resources become interesting subjects of scientific researches [102] which aim to minimize cloud resource costs (energy costs, installation costs, etc.) while respecting their cloud service level agreements (SLAs).

Optimization problems focus not only on the network objectives but also on their parameter details and their intermediate goals including placement restrictions [103], security, full deployment, migration costs [104], anti-collocation (fault tolerance), performance of cloud applications, energy saving, applications' response time [105], revenue maximization, server consolidation [105], etc. As an example, an optimized planning of a virtual network can permit to maximize the total profit of VMs assigned to servers while meeting the placement constraints and server capacity [106].

As virtual networks are evolutions of physical networks that support VMs services, they share many constraints with some other network types; for this reason, the cost model of this paper and its tabu search heuristic is derived not only from related work on virtual networks but also on other network types such as mobile networks [107].

Many studies have recently proposed virtual machines consolidation to construct good quality and efficient data centers [101] [108]. Some of these works [101] [108] may not be suitable for large-sized data centers because of their high execution time. In this study, we focus on the planning

phase of virtual network using a tabu search heuristic suitable for large-sized virtual networks with interdependent VMs.

4.3 System Architecture and Formulation of the Problem

Optimized virtual network planning is important to reduce installation costs of its nodes and to maintain quality of its services. A poorly planned network having interdependent virtual machines (VMs) may not allow efficient data exchange between interdependent applications hosted by these VMs. Oversizing a network planning can help to improve the QoS of a network having interdependent VMs, but it costs very high compared to an optimal solution. This study presents a tabu search (TS) heuristic tool minimizing the cost of deploying a network with interdependent VMs. This TS heuristic aims to approximate the mathematical model defined in [96] which finds an optimal planning cost of a virtual network while respecting delay constraints of the information exchanged among its interdependent VMs. The objective of this TS heuristic is to find quasi-minimal cost of the installed nodes and their links while respecting the constraints of interdependent applications.

Network links and nodes can be of different types characterized by their capacities. Network nodes are virtual machines (VMs), virtual machine managers (VMMs), servers of network data shares (SDSRVs), memory routers (MRs), and hard-disk data routers (HRs). VMM nodes are servers with management applications for virtual machines. SDSRV nodes contain servers with network shared storages for VMs. MR nodes are routers which enable routing memory applications executing real-time data, and HR nodes enable routing virtual hard-disk data (vHDs) or network shared data. In other words, the assignment problem consists of determining VMMs, SDSRVs, MRs, and HRs assignment pattern minimizing a cost function while respecting particular constraints, especially the limited capacities of network nodes.

In our model, virtual network nodes have different potential sites (with x and y coordinates) where they can be placed (see Figure 4.2 and Figure 4.3).

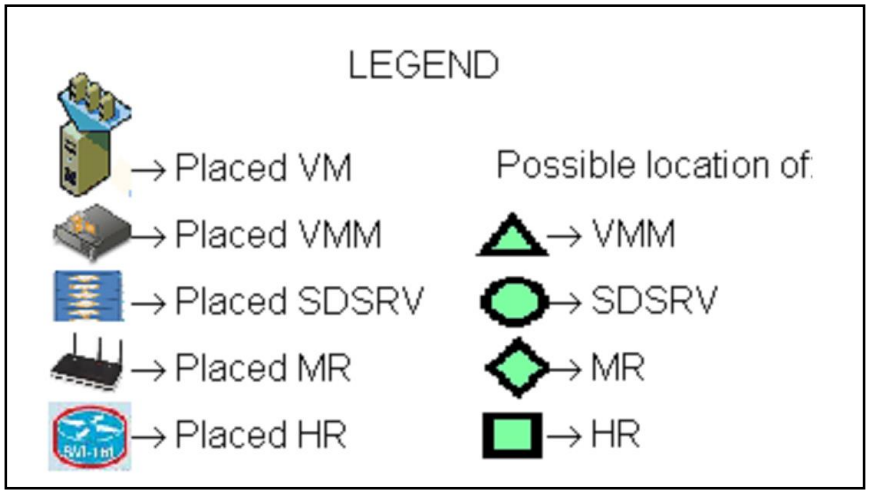

Figure 4.2: Description of the graphical elements used in Figure 4.3 and Figure 4.4.

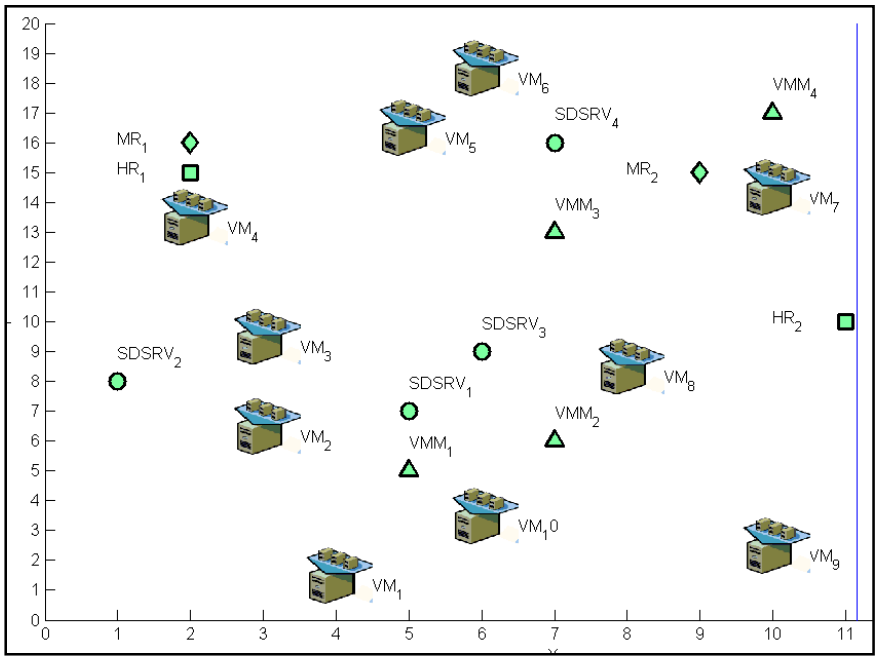

Figure 4.3: An example of initial configuration of a network with interdependent virtual machines.

Figure 4.4 shows a VMs assignment solution where each VM is linked to the VMM_2, the VMM_2 is linked to the $SDRV_4$, and the $SDRV_4$ is linked to MR_2 and HR_1.

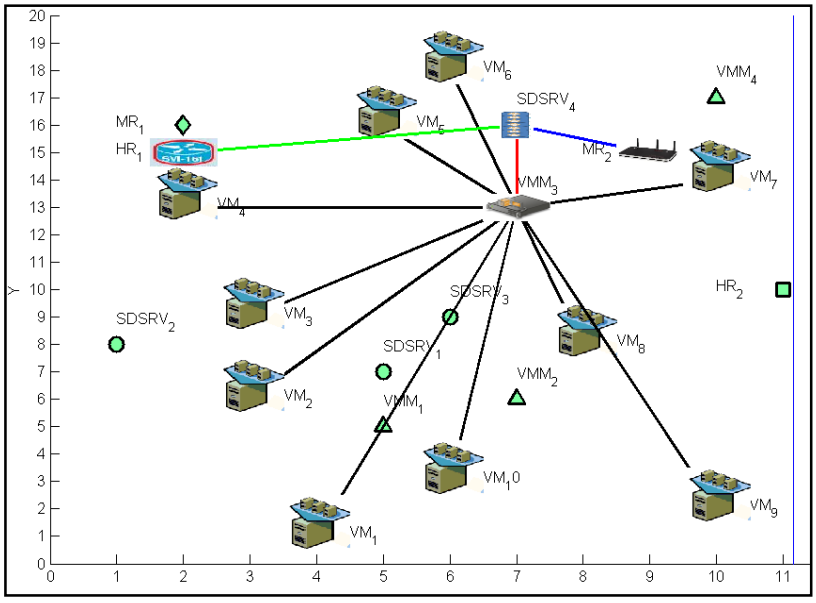

Figure 4.4: An example of assignment solution of interdependent virtual machines

In this paper, our objective function, which is the total cost f (4.1) (more detailed as (6) in [96]) of a virtual network nodes' assignment, is expressed as follows:

$$min\ C_l(v)\ +\ C_n(x) \qquad (4.1)$$

The first term $C_l(v)$ of the equation (4.1) defines the total cost of the links between virtual network nodes (VMMs, SDSRVs, MRs, HRs). The second term $C_n(x)$ represents the cost of the placed nodes. It is interesting to mention that an eventual weighting could be considered in the link and node cost definitions. In addition to this cost function f (1) (more detailed as (6) in [96]), our tabu search heuristic, proposed in this paper, takes into account the delay constraints (detailed as mathematical formulas (1) – (5) in [96]) between interdependent VMs. Due to space limitations, in this paper, we only refer to these constraints which are more detailed in [96].

In this paper, we use an adaptive technique called tabu search (TS) used generally to solve NP-hard problems [109]. The TS basic principle is to find a set of feasible good solutions starting from an initial solution. TS heuristics use generally same techniques to solve problems; however, they differ from one problem to another by their moves to look for neighborhoods of a local solution and by their ways to build and update their tabu lists [109] [110].

In this section, we propose a TS algorithm for the global planning problem of VMs networks

(GPV). Our tabu search (TS) looks for a better solution in the neighborhood of a current solution. This neighborhood is composed of solutions found by applying some moves (transformations from a current solution). In our tabu heuristic, a move is obtained by removing or adding virtual network nodes and links or by modifying their types. To avoid local minimums, our TS uses an aspiration criterion which removes a move from the tabu list when it allows finding better solution than any other solution found so far. Our TS accepts sometimes some solutions which degrade the objective function value f (1) (more detailed in [96]). Also, our TS considers some solutions as "tabu" to avoid cycling with some solutions (see Figure 4.5).

This paper assignment problem consists of minimizing f (1) (more detailed as (6) under constraints (1) – (5) in [96]). This problem is NP-hard and cannot be solved with a standard method such as enumerative searches which are inappropriate for large-sized VM networks. Since they try to find all the possible solution before choosing an optimal solution, they are only efficient for small-sized instances of the problem. For example, for a network composed of m VMMs, n SDSRVs, p MRs, and q HRs, there are "$p * q * m * n$" possible solutions to be examined. Thus, the tabu search, presented in this paper, should be a good tool to find fast acceptable solutions for large-sized virtual network planning.

4.4 Findings and Analysis

Our experiments are executed on forty four instances of our problem generated randomly. We choose randomly the location of VMs and potential locations of the virtual network nodes (VMMs, SDSRVs, MRs, and HRs). Thus, in each instance, a network, with VMs, VMMs, SDSRVs, MRs, and HRs, is used as a sample and it has interdependent VMs sharing data every second.

For the purpose of data interpretation and analysis, we calculate the delay to find a solution with our tabu search (TS) and the total cost of the network including the cost of the placed nodes and the placed links. Thus, we compare these TS performance with those of the mathematical model presented in [96] and solved with CPLEX [111] (see Figure 4.6 and Figure 4.7). Due to space limitations, in this paper, we only show results of a certain number of our test scenarios. Through the results (see Figure 4.6 and Figure 4.7), it can be clearly identified that the tabu search (presented in this paper) finds faster and quasi-equal optimal planning solutions than the exact mathematical model solved by CPLEX [111] for a network with interdependent virtual machines.

- Step 1 (Finding an initial solution) :
  - ✓ Find, as follows, an initial solution solving the assignment problem GPV respecting delay constraints (detailed in [96] as mathematical formulas (1-5)):
    - ➢ For each possible site of VMs, VMMs, SDSRVs, MRs and HRs, put respectively a VMM, SDSRV, MR and HR which types have maximal capacities.
- Repeat the following steps 2 and 3 for "$maxIterations$" times:
  - ✓ Step 2 : Repeat the following steps (2.1 and 2.2) for "$maxNeighbours$" times:
    - ➢ Step 2.1 (Exploring the neighborhood) :
      - ❖ 2.1.1. Find the best move, using the possible tabu moves and aspiration criteria. For each move, find a solution of GPV respecting its delay constraints (detailed in [96] as mathematical formulas (1-5)).
      - ❖ 2.1.2. Define a number of iterations for which the chosen move is tabu (numbers are chosen using a uniform distribution).
    - ➢ Step 2.2 (Updating the TS best solution): If the cost of the current solution is less than the cost of the best solution found so far, the current solution becomes the best solution.
  - ✓ Step 3 (Multi-start best solution updating) :
    - ➢ If the cost of the current solution is less than the cost of the best solution found so far, the current solution becomes the best solution.

Figure 4.5: Algorithm of our tabu search where "$maxIterations$" and "$maxNeighbours$" are input parameters.

Our tabu search results are generally "good" feasible solutions obtained within a reasonable amount of time. The results obtained show that the TS heuristic produced solutions that were, on average, at 0.61% of the optimal solution, and in the worst case at 4.96% of the optimal solution. As a result, these TS results are excellent since the planning problem of virtual networks is a NP-problem difficult to solve for large-sized networks with exact methods.

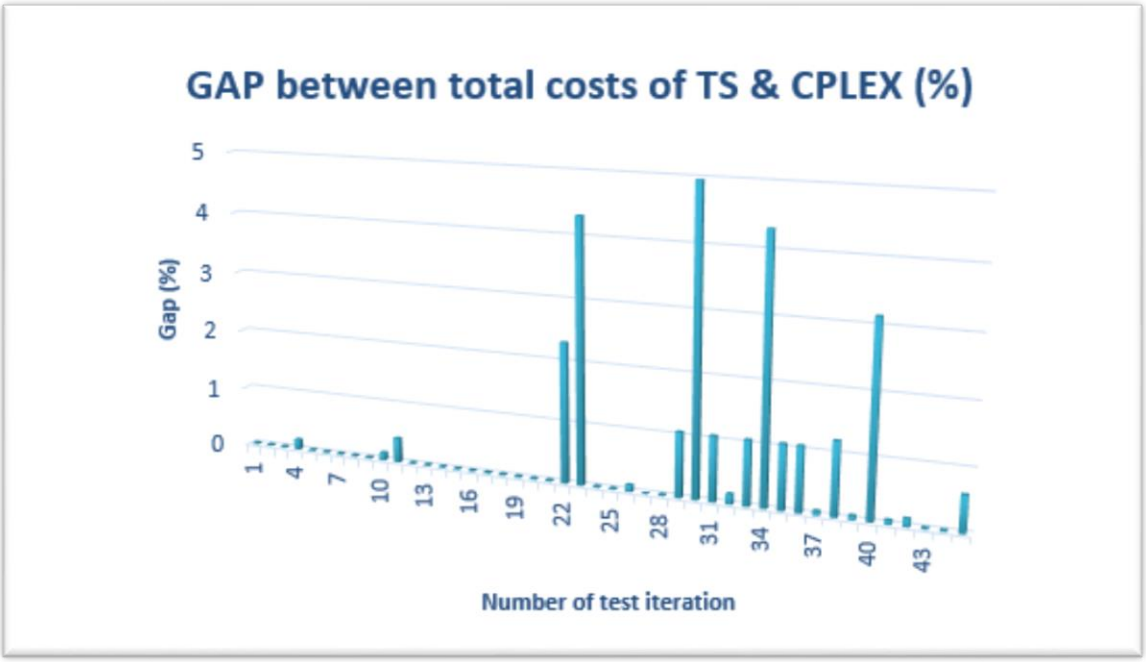

Figure 4.6: Gap between the total cost, of a simulated virtual network, of TS and CPLEX.

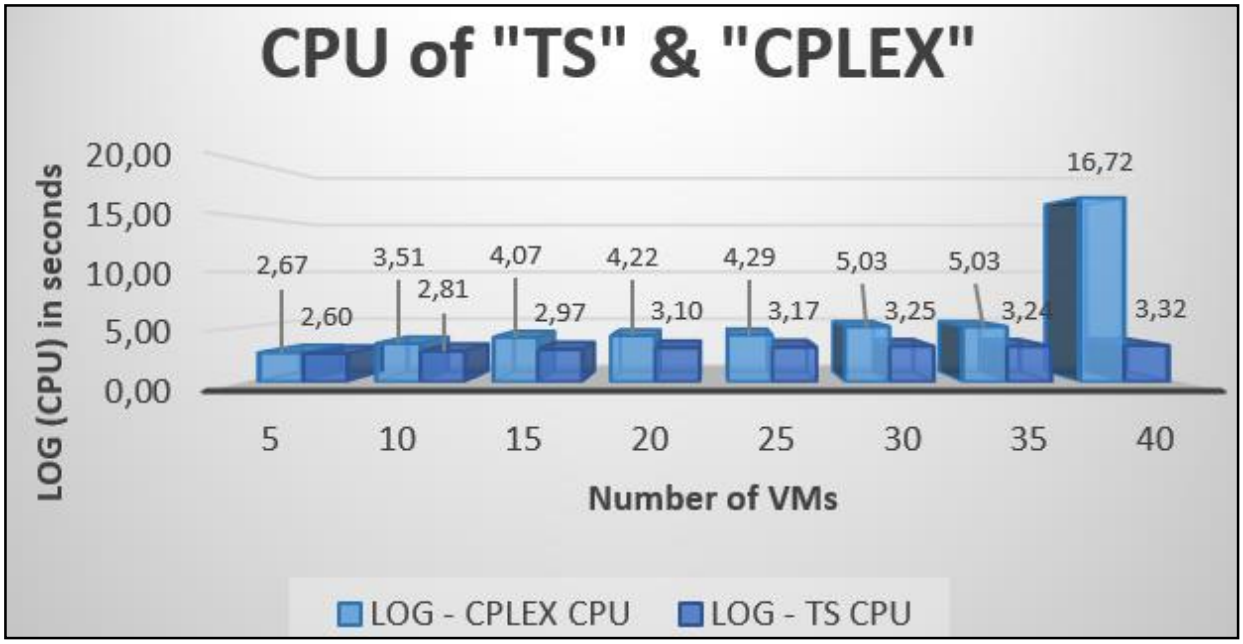

Figure 4.7: CPU execution time to find solutions (by CPLEX and Tabu search (TS))

4.5 Discussions

The results of this research study showed that it is possible to plan a quasi-optimal network with interdependent virtual machines. As known, interdependent VMs are those which share data between their hosted applications. Actual planning tools are not suitable for such interdependent VMs and are more useful for non-correlated VMs.

Through the research findings, interdependent VMs share their data with a limited predefined delay in order to satisfy their network QoS. Thus, these interdependent VMs communicate their data to other VMs which may need these data to properly continue their service executions. For this reason, these data exchanges should be done under a short limited delay.

Also, this indicates that a random VM planning may not permit a good data exchange between interdependent VMs. Thus, planning tools should emphasis more on data exchange delays to enhance network quality

For small networks, it is possible to find such network with optimal planning for interdependent VMs and shared resources; however, in a large-sized network, it is quasi-impossible to find quickly a good VM planning. Thus, when designing VMs' planning strategies, planning tools should take into account the data exchange constraints of interdependent VMs. The tabu search heuristic delivered in this paper considers these delay constraints to deliver a virtual network planning with good QoS due to well data exchange among its interdependent VMs. One of the strength of the tabu search heuristic, presented in this paper, is its use to find quasi-optimal cost for network planning in an acceptable delay especially for large-sized virtual networks. However, the research of this paper is still in its infancy state that could be more explored further to consider many other constraints such as the overall capacities of physical machines and of the virtual machines co-located on them.

4.6 Conclusion

This paper presents a preliminary research study to define a tabu search heuristic of the problem of planning of interdependent virtual machines in a data center. The research results indicate that it is possible to find a near-optimal planning for a virtual network where interdependent VMs can exchange their data efficiently.

Various planning approaches, including other types of heuristics, can find faster acceptable planning solutions which find better quasi-optimal solutions than the one found by this paper tabu search heuristic.

CHAPITRE 5 ARTICLE 2: REAL-TIME QOS ISSUES IN LIVE MIGRATION OF INTERDEPENDENT VIRTUAL MACHINES

Auteurs : Salah-Eddine Benbrahim, Alejandro Quintero and Martine Bellaiche

IEEE - 2014 8th Asia Modelling Symposium (AMS)

Abstract

This paper, based on papers [91] [92] [93], presents and develops a mathematical model to meet the strict QoS constraints of (soft) real-time virtualized distributed applications while their hooting interdependent Virtual Machines (VMs) are undergoing a live migration. To this purpose, it is vital to take simultaneously into account the network resource requirements of a VM live migration, the data center SLAs contracts, and the quality of services of migrating interdependent applications. Set of simulations are executed and show that it is possible to respect SLAs contracts in a data center and to maintain QoS of interdependent applications while their hosting VMs undergoing live migrations.

5.1 Introduction

Virtual machines (VMs) can be placed and moved from physical machines (PMs) to others as they are loosely coupled to physical data center infrastructures. VM live migrations can help to adjust dynamically and automatically resources in a data center. This resource management can be achieved in a centralized or decentralized [50] manner by defining a utility function taking into account the levels of the resource utilization.

Checconi et al. [91] presented an effective technique for live migration of real-time independent virtualized applications which achieves very low and predictable down times. In this paper, we propose a new live migration approach based on the paper [91] and extended to interdependent VMs while the paper [91] focused only on non-correlated VMs.

This paper proposes a model formulation to efficiently solve the problem of live migration of interdependent VMs in virtual networks with regards to SLAs violations. Section 5.2 provides background information and related work. Section 5.3 describes our mathematical formulation for the proposed approach of interdependent VM live migration. Section 5.4 deals with the experimental results. Finally, Section 5.5 discusses the results while Section 5.6 concludes this study with some possible further work.

5.2 Overview of Cloud Networks and Virtual Machine Live Migration Techniques

Many cloud computing products, such as IBM Blue Cloud [100] and Amazon EC2, provide to end users access to web applications. Thus, computing applications, hosted by these Cloud networks, are just-in-time and transparent services to the user. Cloud computing services are categorized generally as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [101].

As they are flexible, scalable and easy to maintain, the cloud platform have an increasing popularity. Thus, there are many interesting research subjects [102] on performance management of the cloud resources and their scheduling. These researches aims to maximize the overall network quality, to respect service level agreements (SLAs), and to minimize the resources costs (energy costs, installation costs, etc.).

Live migration of distributed and shared resources in a data center can solve the problem of resource under-utilization. Also, it can be a good solution to solve the problem of slow services sometimes due to bad allocation of interdependent VMs, which may lead to violations of service level agreements (SLAs). These SLA violations can be severe unless live migration solutions take into account the data propagation delay between interdependent VMs, down-time of their services and the limited number of concurrent VM migrations in a network.

An effective technique for independent VMs is addressed in [91] to meet timing constraints of real time virtualized applications hosted by VMs undergoing a live migration. This technique [91] identifies, reserved and temporally isolated appropriate shared resources; then, it [91] applies a proper scheduling algorithms (for both CPU and network) to reserve resource shares to individual VMs. Checconi et al. [91] tested this technique by patching KVM at the kernel level, and their obtained results highlight the benefits of the proposed approach in the case of independent VMs; however, this technique seems to be less efficient for interdependent VMs because it does not take into account delay constraints between interdependent applications hosted by VMs undergoing live migrations.

Clark et al. [10] introduces a writable working set pre-copy approach where a Virtual Machine Monitor (VMM) has to migrate a VM and all of its associated resources (ex: memory, internal state

of devices, and virtual CPU) to a new PM. Generally, the most time-consuming resource to transfer is the VM memory. Checconi et al. [91] used a technique called pre-copy where VMM transfers memory pages while VM is running, and it checks dirtied pages after every transfer step in order to retransmit them. This process is executed for many periods until a final one. In this final period, VM is stopped, remaining dirtied pages are transmitted and the VM is restarted on the destination. In this study, we propose an extended writable working set pre-copy approach which is useful for interdependent VMs live migration and which is based on the general writable working set pre-copy approach described in [10] and [91].

In this study, we focus on the live migration of interdependent VMs minimizing the SLA violations and the total live migration delays.

5.3 System and Model Formulation of the Problem

In this section, we present a SLA-aware real-time live migration for interdependent VMs in a cloud computing system which aims to minimize the total migration cost of the system. To increase paper readability, we first describe the key symbols used throughout this paper along with their definitions (see Table 5.1).

Table 5.1: Notation and definitions

Input/output	Symbol	Definition
	i	Index of a VM in a data center.
	j	Index of a resource in a data center.
	l	Index of a service in a data center.
input	$Max_{transDelay}$	Maximal acceptable total delay of pre-copy phase of a VM migration.
input	$Max_{transPageDelay}$	Maximal acceptable delay of pre-copy phase of a single migrated page.
input	$Max_{length,mappedPage,i}$	Maximal size of a mapped page of a VM_i.

Input/output	Symbol	Definition
input	$Max_{number,mappedPage,i}$	Maximal number of mapped pages of a VM_i.
input	$Max_{length,transmittedPage,i}$	Maximal length of transmitted pages of a VM_i.
input	$Max_{number,transmittedPage,i}$	Maximal number of transmitted pages of a VM_i.
input	$Max_{length,downPage,i}$	Maximal length of down-pages of a VM_i.
Input	$Max_{number,downPage,i}$	Maximal number of down-pages of a VM_i.
Input	$Max_{dependRespTime,i}$	Maximal acceptable delay of communication between a VM_i and its interdependent VMs.
Input	$Max_{cpuMigr,i}$	Maximal CPU amount reserved for migrating a VM_i.
Input	$Max_{finalCopyBW}$	Maximal bandwidth amount reserved for the final copy of a VM_i migration.
Input	$Max_{transCopyBW}$	Maximal bandwidth amount reserved for the pre-copy of a VM_i migration.
Input	Max_{header}	Maximal header length of a transmitted page.
Input	RES(Data center)	Resource set of a data center.
input	SRV(Data center)	Service set of a data center

Input/output	Symbol	Definition
input	ID(VM_i)	Set of interdependent VMs of a VM_i.
input	$SLA_{cpuExec,l}$	SLA on the minimal CPU amount for a service l performed on a VM.
input	$SLA_{latency,l}$	SLA on the latency for a service l performed on a VM.
input	$SLA_{respTime,l}$	SLA on the response time for a service l performed on a VM.
output	$T_{trans,i}$	Total delay of pre-copy migration of a VM_i.
output	$T_{down,i}$	Down-time of migration of a VM_i.
output	$T_{latency,l}$	Latency of a service l performed on a VM.
output	$T_{tranPage,i}$	Delay of transmitted page of a VM_i.
output	$T_{dependRespTime,l}$	Communication delay between a service l hosted by a VM and its interdependent VMs.
output	$B_{transCopy,i}$	Network bandwidth (Ex: 50 Mbit/s) reserved and used for pre-copy migration of a VM_i.
output	$B_{finalCopy,i}$	Network bandwidth (Ex: 50 Mbit/s) reserved and used for final-copy migration of a VM_i.

Input/output	Symbol	Definition
output	$D_{cpuMigr,i}$	CPU amount used for the migration of a VM_i.
output	$D_{cpuExec,l}$	CPU amount used by a service l performed on a VM.
output	$P(Res_j)$	Penalty of a resource j.
output	$Q(Srv_l)$	Quality of a service l (ex: response time to queries, etc.) adhering to specific SLAs.
output	$Resp_l$	Response time of a service l.
output	$H_{tranPage,i}$	Run-time overhead of a VM_i.
output	$L_{mappedPage,i}$	Length of mapped pages of a VM_i.
output	$N_{mappedPage,i}$	Number of mapped pages of a VM_i.
output	$L_{transmittedPage,i}$	Length of transmitted pages of a VM_i.
output	$N_{transmittedPage,i}$	Number of transmitted pages of a VM_i.
output	$Prob_{retransmit,dirtyPage,i}$	Estimated probability of retransmitting a migrated VM_i page (based on historical data, stochastic model, etc.)
output	$L_{downPage,i}$	Length of down-pages transmitted in the final stage of the migration of a VM_i.
output	$N_{downPage,i}$	Number of down-pages transmitted in the final copy stage of the migration of a VM_i.

Our problem has a multi-purpose function as it aims to maximize the total service quality of a data center and minimize the total live migration delay of a VM, its final stop delay and the total violations of data center SLAs. Then, this multipurpose function is defined as:

$$\begin{cases} \min T_{down,i} & \forall i \quad (5.1) \\ \min T_{trans,i} & \forall i \quad (5.2) \\ \min \sum_j P(Res_j) & \forall j \quad (5.3) \\ \max \sum_l Q(Srv_l) & \forall l \quad (5.4) \end{cases}$$

where:

- Equation (5.1) aims to minimize the stopping delay (down-time) of a migrated VM.
- Equation (5.2) aims to minimize the total time of the pre-copy stage of a VM live migration.
- Equation (5.3) aims to minimize the total of the SLAs violations for the entire data center.
- Equation (5.4) aims to maximize the total sum of service qualities for the entire network.

This multi-purpose function (5.1) - (5.4) is subject to the following constraints (5.5) - (5.22):

$$T_{down,i} < SLA_{respTime,l} \quad \forall i,l \quad (5.5)$$

$$T_{trans,i} < Max_{transDelay} \quad \forall i \quad (5.6)$$

$$T_{latency,l} < SLA_{latency,l} \quad \forall i \quad (5.7)$$

$$T_{tranPage,i} < Max_{transPageDelay} \quad \forall i \quad (5.8)$$

$$H_{tranPage,i} < Max_{header} \quad \forall i \quad (5.9)$$

$$B_{finalCopy,i} < Max_{finalcCopyBW} \quad \forall i \quad (5.10)$$

$$B_{transCopy,i} < Max_{transCopyBW} \quad \forall i \quad (5.11)$$

$$Resp_l < SLA_{respTime,l} \quad \forall i \quad (5.12)$$

$$D_{cpuExec,l} < SLA_{cpuExec,l} \quad \forall i \quad (5.13)$$

$$D_{cpuMigr,i} < Max_{cpuMigr,i} \quad \forall i \quad (5.14)$$

$$L_{mappedPage,i} < Max_{length,mappedPage,i} \quad \forall i \quad (5.15)$$

$$N_{mappedPage,i} < Max_{number,mappedPage,i} \quad \forall i \quad (5.16)$$

$$L_{transmittedPage,i} < Max_{length,transmittedPage,i} \quad \forall i \quad (5.17)$$

$$N_{transmittedPage,i} < Max_{number,transmittedPage,i} \quad \forall i \quad (5.18)$$

$$L_{downPage,i} < Max_{length,downPage,i} \quad \forall i \quad (5.19)$$

$$N_{downPage,i} < Max_{number,downPage,i} \quad \forall i \quad (5.20)$$

$$Prob_{retransmit,dirtyPage,i} < Max_{retransmitProba,dirtyPage,i} \quad \forall i \quad (5.21)$$

$$T_{dependRespTime,l} < Max_{dependRespTime,l} \quad \forall l \quad (5.22)$$

where:

- Inequality (5.5) aims to keep the down-time $T_{down,i}$ of a VM_i below a response time specified in a $SLA_{respTime,l}$ contract defining the lower acceptable response time of a service l to end customers.
- Inequality (5.6) aims to keep the total pre-copy delay $T_{trans,i}$ below the threshold $Max_{transDelay}$ defined by data centers.
- Inequality (5.7) aims to keep latencies $T_{latency,l}$ of a service l below a threshold defined by the $SLA_{latency,l}$ contract.

- Inequality (5.8) aims to keep transmission delays $T_{tranPage,i}$ of a VM_i below a threshold $Max_{transPageDelay}$ defined by data centers.
- Inequality (5.9) aims to keep overheads of transmitted pages $H_{tranPage,i}$ below a threshold Max_{header} defined by data centers.
- Inequality (5.10) aims to keep bandwidths used for final copy $B_{finalCopy,i}$ below a threshold $Max_{finalCopyBW}$ defined by data centers.
- Inequality (5.11) aims to keep bandwidths used for pre-copy $B_{transCopy,i}$ below a threshold $Max_{transCopyBW}$ defined by data centers.
- Inequality (5.12) aims to keep response times $Resp_l$ of a service l below a threshold $SLA_{response,l}$ defined by data centers.
- Inequality (5.13) aims to keep CPU amounts $D_{cpuExec,l}$ used for the execution of a service l below a threshold $SLA_{cpuExec,i}$ defined by data centers.
- Inequality (5.14) aims to keep CPU amounts $D_{cpuMigr,i}$ used for the migration of a VM_i below a threshold $Max_{cpuMigr,i}$ defined by data centers.
- Inequality (5.15) aims to keep lengths $L_{mappedPage,i}$ of mapped pages for the migration of a VM_i below a threshold $Max_{length,mappedPage,i}$ defined by data centers.
- Inequality (5.16) aims to keep numbers $N_{mappedPage,i}$ of mapped pages for the migration of a VM_i below a threshold $Max_{number,mappedPage,i}$ defined by data centers.
- Inequality (5.17) aims to keep lengths $L_{transmittedPage,i}$ of transmitted pages for the migration of a VM_i below a threshold $Max_{length,transmittedPage,i}$ defined by data centers.
- Inequality (5.18) aims to keep numbers $N_{transmittedPage,i}$ of transmitted pages for the migration of a VM_i below a threshold $Max_{number,transmittedPage,i}$ defined by data centers.
- Inequality (5.19) aims to keep lengths $L_{downPage,i}$ of down-pages for the migration of a VM_i below a threshold $Max_{length,downPage,i}$ defined by data centers.
- Inequality (5.20) aims to keep numbers $N_{downPage,i}$ of down-pages for the migration of a VM_i below a threshold $Max_{number,downPage,i}$ defined by data centers.

- Inequality (5.21) aims to keep estimated retransmission probabilities $Prob_{retransmit,dirtyPage,i}$ of dirty-pages of a VM_i below a threshold $Max_{retransmitProba,dirtyPage,i}$ defined by data centers.
- Inequality (5.22) aims to keep response times $T_{dependRespTime,l}$, between a service l hosted by a migrated VM and its interdependent VMs, below a threshold $Max_{dependRespTime,l}$ defined by data centers.

5.4 Findings and Analysis

In order to test our solution, we have built a simulator running in a stepwise manner and where resource requirements evolve and VM live migrations are executed. Thus, this test environment permit to test different methods with different scenario (involving varying numbers of VMs and physical machines (PMs) with the same conditions. In this simulator, PMs are represented as resource bins hosting interdependent VMs which are implemented as interdependent units with varied length of busy and idle periods in order to generate a system-wide with similar behavior that is generally observed in online data centers [112]. Due to space limitations, in this paper, we only show some result of some test scenarios (see Figure 5.1).

In this simulator, we use a network, with 2,500 PMs and 25,000 VMs, as a sample where every PM hosts 10 VMs. This network has interdependent VMs placed on different PMs and share data every second. For the purpose of data interpretation and analysis, we calculate network SLAs penalties, VM live migration down-time and its pre-copy delay. The results are shown in Figure 5.1.

Through results (see Figure 5.1), it can be clearly identified that the formulation model (presented in this paper for interdependent VMs) finds better results than the general pre-copy technique defined in the paper [10]. Thus, in our simulated data center, the number of SLAs violation of our approach is lower than the general pre-copy technique [10].

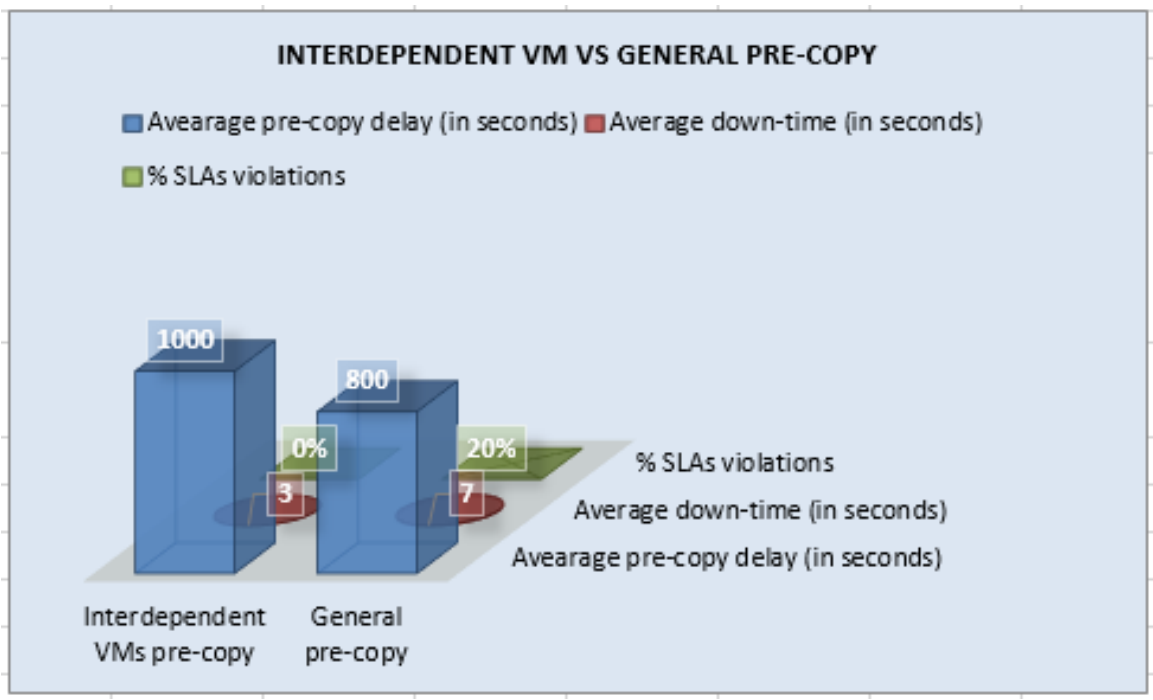

Figure 5.1: A view of our interdependent VMs pre-copy approach vs general pre-copy [10] in terms of the number of SLAs violations, pre-copy and down-time delays.

5.5 Discussions

The results of this research study showed that it is possible to find techniques respecting SLAs on QoS of applications hosted by interdependent VMs undergoing a live migration in a data center. As known, interdependent VMs are those VMs that share data between them with short predefined delay. The actual migrating tools may limit their approaches just for networks with independent VMs. Interdependent VMs need tools who permit them to transmit and propagate their data on the network where other VMs may need to receive these data before continuing executing their services. Thus, these data exchange must be finished under a short limited response time.

Also, this indicates that a pre-copy or a post-copy VM migration technique may not permit a good data exchange between interdependent VMs undergoing a live migration in a data center. Thus, new live migration tools should emphasis more on the data exchange to enhance the network quality. It is suggested to live migrate interdependent VMs in a way that they can continue to communicate quickly. For small networks, it is possible to find such tool with optimal live migration for interdependent VMs; however, in a large-sized network, it is quasi-impossible to find quickly such

a good VM live migration tool. Thus, in the design of VM live migration strategies, the migration tools should be aware of the data exchange constraints of interdependent VMs and of other QoS constraints defined in SLAs contracts in a data center. The formulation model presented in this paper takes into account theses constraints to deliver a good VM live migration where the quality of the network depends on a well data exchange among its interdependent VMs. One of the strength of the formulation model, presented in this paper, is the increasing of the quality of migrating services while decreasing the total live migration delay in a virtual network. However, this paper research is still in its infancy state and could be more explored further. For example, our mathematical model execution time is subject to improvement especially for large-sized data centers in order to use less CPU and memory. Also, in further works, it is useful to implement and test this paper formulation model in a decentralized virtual networks.

5.6 Conclusion

This paper presents the preliminary research study to define a model formulation of the problem of maintaining the QoS of applications hosted by interdependent VMs undergoing a live migration in a data center. The research results indicated that it is possible to find a technique where an interdependent application can maintain its QoS while its hosting VM is migrating and can continuously exchange its data efficiently with its interdependent virtual machines. Heuristics and other various approximation approaches can be used to find faster acceptable solutions to approximate the model formulation described in this paper especially for large-sized virtual networks.

CHAPITRE 6 ARTICLE 3: LIVE PLACEMENT OF INTERDEPENDENT VIRTUAL MACHINES TO OPTIMIZE CLOUD SERVICE PROFITS AND PENALTIES ON SLAS

Auteurs : Salah-Eddine Benbrahim, Alejandro Quintero, *Member, IEEE*, and Martine Bellaiche

Submitted to : IEEE TRANSACTIONS ON SERVICES COMPUTING

Abstract

This paper aims to optimize cloud services' penalties and net profits. This optimization is a complex task due to the difficulty of achieving a successful compromise between penalties on service level contracts and live placement of interdependent virtual machines (VMs). This paper studies this optimization problem to minimize services' penalties while achieving live migrations of interdependent VMs. This VMs' live placement optimization problem is NP-hard with exponential running time, especially for large-sized instances; for this reason, we approximate it with efficient heuristics. We test our formulation and heuristics for cloud services where the overall services' penalty needs to be minimized and where efficient live migrations of VMs is a concern. We present extensive simulations with services hosted by VMs and physical machines (PMs) to demonstrate the efficiency of our heuristics. Our results show how cloud service providers may live place their profitable services with live migrating VMs. Finally, our results show that our heuristics: (i) find better solutions for the existing machines' configuration of Google traces; (ii) are suitable for large-sized instances of cloud services with tens of thousands of PMs and VMs; and (iii) perform better than the benchmark of Google traces in terms of overall penalties and profits.

6.1 Introduction

To deliver services to many users using different types of computers, more and more service providers use cloud servers to serve their clients. These cloud services are supported by many companies and products such as IBM Blue Cloud [100] and Amazon EC2 [113]. Many market researches, such as Forbes, predict that the cloud subscription revenue will increase to 106 billion USD by 2016 [114], and some leading IT companies, such as Cisco [115], have already become big players in this market opportunity; for this reason, we notice many big companies using and providing cloud services. However, providing cloud services is very demanding, as their net profits

is negatively correlated to necessary investment on materials; for example, technical problems are very expensive as described by the study "Top 5 AWS EC2 Performance Problems" [116]. Therefore, one of principle challenges for cloud providers is to decrease the overall services' penalty while reducing the hardware investment in physical machines. Cloud client satisfaction demands, for less-end penalty on service level agreements (SLAs), may request high financial investment on materials; however, using low-end material resources leads to less acceptable quality of cloud services, and it may cause profit decreasing for cloud providers. Furthermore, different service types demand different hardware requirements, which may lead to wasted or overused hardware resources if physical and virtual resources are not well planned and/or not well load-balanced. For example, servers configured for intensive real-time requests may be not suitable for occasional off-line requests. The real-time service requirement diversity makes the dilemma of finding the best compromise between the penalties of services and cloud provider profits more difficult. As cloud services are hosted by virtual machines hosted themselves by physical servers, server consolidation becomes an important research field. This server allocation helps cloud IT managers to dynamically allocate resources among cloud servers used by various end-users for lower overall services' penalties and minimal-enough operational costs. In this paper, we study the problem of live consolidating cloud services on physical machines using live migration of interdependent virtual machines (VMs) in order to provide less penalties on SLAs at good-enough operational costs. We study the VM live placement problem as a technique to minimize the total service penalties while providing the just-good-enough live migrations of interdependent virtual machines. These VMs' live placement problem is defined as penalty-centric problem in this paper. This problem of virtual machine live placement is a variation of the NP-Complete virtual network embedding problem [117] solved approximately by [117] [118] [119] [120] [121]; however, these solutions are more suitable for applications with high CPU/disk throughput than for cloud intensive applications demanding less penalties on SLAs and for real-time resources requirements [122] [123] [124] [125] with live migrating virtual machines. Particularly, dissimilar to applications that request high CPU/Bandwidth, cloud services with live VM migrations demand low penalties on contract level agreement (SLAs) [123]. Hence, the existing virtual network solutions do not fit for cloud services hosted by live migrating interdependent virtual machines. This paper, to the best of our research, is the first to propose a solution for the problem of live VM placement minimizing cloud service penalties with live VM migrations.

In particular, this paper outlines the following contributions:

- We formulate and propose an efficient and practical heuristic for penalty-centric VM live placement problems with live migrating interdependent virtual machines. This live migration refers to moving a running VM from a physical machine to another (see Section 6.4).
- Our trace-driven simulations show that: (i) our efficient heuristic results have better performance than other benchmarks; (ii) our efficient heuristic fits for large-sized instances of cloud services with many interdependent virtual machines and physical machines; and (iii) our efficient heuristic performs better than existing placement solution in an acceptable delay (see Section 6.7).

6.2 Related Work

6.2.1 General Cloud Applications

As seen in [126], many studies have discussed optimizing problems of cloud services. Some of these studies, such as Zaman and Grosu [127], present VM dynamic allocation techniques to optimize the cloud provider's profit while maximizing the total utilization of resources. In our study, we formulate the VM live placement problem of cloud services and we develop optimization heuristics to solve this NP-Complete problem. Contrary to Zaman and Grosu [127], we study the VM live migration with the objective of minimizing the overall penalty of cloud services through VM interdependency-aware algorithms while simultaneously optimizing the VM live placement.

As seen in [126], live VM migrations for non-real-time cloud services have been discussed in many studies such as Marzolla et al. [128], Ferreto et al. [129], and Speitkamp and Bichler [8]. These studies have used dynamic algorithms to consolidate physical servers and put empty servers to sleep to minimize their energy consumption and reduce the number of necessary live migrations. Some other studies, such as Chen et al. [130] and Nathuji et al. [131], use migration history to save energy consumptions, find appropriate resource bounds, and consolidate services for better QoS. None of these studies have considered the penalties on service response time while live migrating VMs.

6.2.2 Virtualization

As discussed in [126], various virtualization technologies have been proposed in the literature and used in data centers; these technologies, described by various literatures such as [132], [133], [134], and [59], are categorized in six groups: (i) application virtualization (AV) such as Wine and java virtual machine (JVM), which can execute java programs on various operating systems (OS); (ii) resource virtualization (RV), such as Gluster, which is a virtualization technique for hosted resources such as CPU, network, and memory; (iii) operating system level virtualization (OSLV), such as OpenVZ, which can run many guests on a single OS and can dynamically manage resources; (iv) para-virtualization (PARA), used only with open source OS, which can send communication directly from guest operating systems to hardware; (v) hardware virtual machine (HVM), such as VirtualBox, also known as virtualization with hardware assistance, which help guests send their communication directly to hardware; and (vi) full virtualization (full), using virtual machine monitor (VMM), which can support many guests without regard to their OS types. These six virtualization categories can be divided into two main groups: (i) virtualization techniques, including AV, RV, and OSLV, which enable running various guests on the same OS; and (ii) virtualization techniques, including PARA, HVM, and full, which isolate guests by running each one on a different OS.

As described in [8], virtualization can be controlled by a program responsible for creating "guest" software hosting applications and their OS. There are many virtualization products such as HP nPAR and IBM DLPAR known for their symmetric multiprocessor (SMP) servers running different OS; also, there are hypervisor software products which can run simultaneously secure and isolated virtual servers on a single physical machine. In addition, these hypervisors can monitor and support live migrating of VMs [135]; however, this virtualization causes overheads because of necessary additional CPU cycles [126].

6.2.3 Server Consolidation

As described in [8], server consolidation is a technique that optimizes usage costs of computer server resources. This consolidation approach allows the reorganization of the workloads of different services to minimize the data center operational costs. These costs are due mainly to energy consumption, server purchasing, server administration, and cloud center maintenance. As

described in [8], this server consolidation problem focuses mainly on minimizing server costs (purchasing, maintenance, administration, etc.). This minimization is based on efficient characteristics and the number of the potential servers to host data center services efficiently while respecting technical constraints.

According to [8], there are three different scenarios, from the point of view of data center providers, where consolidation problems should be optimally solved: (i) the first scenario is an investment decision where a data center manager wants to evaluate and acquire new generations of materials and software to deliver virtual services; in this situation, the responsible party needs to choose which types of servers to purchase according to their characteristics and their costs (including energy and administrative costs, etc.); thus, an optimization problem, with objective function of minimizing the total cost of servers, allows an efficient server's configuration to be found with strategic considerations such as hardware reusability; (ii) the second scenario, called the "sunk costs" context where materials have already been purchased, is an optimization problem with objective function of minimizing operational costs including mainly server administration, electrical power supply for servers and their cooling costs; and (iii) the third and last scenario, where servers have the same costs and characteristics, is an optimization decision for choosing the right subset of servers to use and the right subset of servers to put to sleep; this optimization problem has the objective function to minimize the number of required servers subject to technical allocation constraints.

As mentioned by Martin Bichler et al. [8], the performance of services in a data center can be measured by some standard metrics; for example, memory may be measured in Gigabytes, bandwidth in Megabits per second, and CPU capacity in HP Computons or SAP Application Performance Standard (SAPS); SAPS is derived from the SAP Sales and Distribution Benchmark [136]. Fortunately, many data centers periodically record their server workloads (CPU, memory, bandwidth utilization, etc.) on real-time; these records show periodical seasonal patterns for each resource utilization; for example, OLAP applications for managers' reporting show a daily peak in the morning while payroll accounting show peaks on the weekends; these two activities are examples of services that are naturally negatively correlated (see Figure 6.1) and so they can be consolidated on the same server. Martin Bichler et al. [8] also indicate that there is not enough scientific work on server consolidation, contrary to works on capacity planning and resource allocation; for this reason, their work focused more on the consolidation of servers having

variations in workload over time, and they proposed heuristics for this optimization problem and for data aggregation for every allocation decision.

6.2.4 Complexity Analysis and Algorithmic Solutions

As developed in [8] and in order to know if an optimization problem can have an exact solution for large instances, we need to start by a complexity analysis. The placement problems of virtual machines, such as this paper's live placement optimization problem, are derived from the basic server consolidation problem known as the Static Server Allocation Problem (SSAP). SSAP solutions allow cloud service managers to consolidate servers by affecting virtual servers to target physical servers in order to minimize the number of required servers and their overall costs. Unfortunately, SSAP and its derived problems, such as this paper optimization problem, are proven to be strongly NP-hard even for problems with only one resource and servers having the same capacities and costs [8]. This SSAP and its derived problems can be reduced to the multidimensional bin packing problem (MDBP) proven as NP-hard by Garey, Graham, Johnson & Yao [137]. Since this MDBP is difficult to solve with polynomial time approximation schemes (PTAS), many studies focused on finding its non-trivial solutions; for example, Chekuri & Khanna [138] propose a PTAS based on a linear programming relaxation for the MDBP problem and they deliver a $(1 + \epsilon * d + O(\log \epsilon^{-1}))$-approximate solution with predetermined constants ϵ and d; another approximate solution was proposed by Bansal, Caprara, and Svirdenko [139], who guarantee an approximation solution close to $ln(1 + d)$ with fixed d for MDBP. The approximate solutions for MDBP, such as the best fit decreasing (BFD) and the first fit decreasing (FFD) approaches [139], cannot always be intractable for practical problem sizes as shown in [140] [141]; thus, it is important for IT managers to know the limits of these solutions in terms of solution quality and problem size. Generally, complexity study results provide the worst-case results and do not give enough helpful information for server consolidation problems; this information about problem size and input values heavily impacts the problem solution quality as discussed in [8].

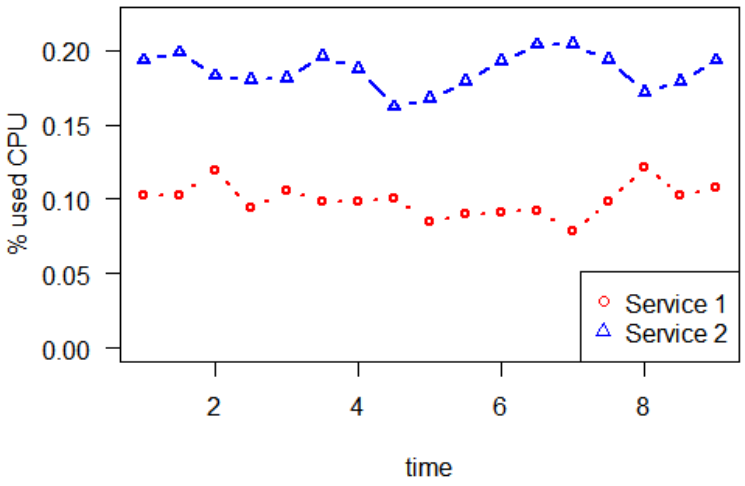

Figure 6.1: Complementary workload of two services of Google data traces from 2011 [95].

6.3 Measurement Studies and Simulation Data

6.3.1 Simulation Data

Many scientific studies, such as [8], obtain workload data sets from their industry partners; these test sets concern periodical traces of resource usage for various types of cloud services such as web/application/database (W/A/D) services and ERP applications. Therefore, these optimization studies use statistical methods, derived from time series analysis, to look for periodical seasonality. This information may help cloud managers to deduce some significant trends in resource demand over time. IT managers should find and periodically use these trends to re-optimize and reallocate their servers [8]. CPU is among the main resources of these servers and it is well known as a bottleneck resource for many applications, especially when the ratio of CPU power to memory size is small [142]; for this reason, in this paper, we consider both CPU and memory in addition to bandwidth as resources contrary to studies conducted by Martin Bichler et al. [8] who considered only CPU in their experiments. It is important to note that resource utilization in some application types, such as ERP services, are usually higher than other group of application types, such as W/A/D services; both of these service group types are tested in our study to find out runtime and quality of solutions obtained by our mathematical models and heuristics (see Section 6.6); furthermore, it is important to notice that our study tests its algorithms on Google data center traces from 2011 [95].

Some other studies use their own measurement studies; for example, Hua-Jun Hong et al. [126] installed a gaming client connected to a virtual server hosting three games; then, they collected different performance metrics over a five-minute period using techniques proposed in Chen et al. [143]. These measures concerned CPU, network and processing delays to treat and serve cloud services to data center clients; these measures show that there is no evidence prediction for application overheads as they depend on server specifications, operating systems and application/VM pairs [126]; therefore, measurement approaches for cloud services hosted on heterogeneous servers should use online regression for continuous adaptations. These offline and online regression models can sometimes be approximated by sigmoid functions if their R-square values approach the value "1" [126] [144]. In our study and contrary to Hua-Jun Hong et al.[126], we can not use sigmoid functions as regression models for Google services [95] as their resource utilizations do not follow sigmoid functions (see Figure 6.2). Time series of simulation data may be obtained from cluster traces from many enterprises such as Yahoo!, Google, and Facebook [145] [146] [147] [148] [149] [150] [151]. Also, simulation data (network delay between server and users, etc.) may be measured by many network diagnostic tools such as Gummadi et al. [152]. In our study, we use Google's data [95] to simulate and test our heuristics.

This paper uses Google traces from 2011 [95] to test our mathematical models and heuristics. Figure 6.3 shows a workload profile from a sample of Google PMs' traces; Figure 6.4 shows a workload profile of a Google task in 2011. Figure 6.5 shows a summary of a sample of Google data traces [95] used by our heuristics.

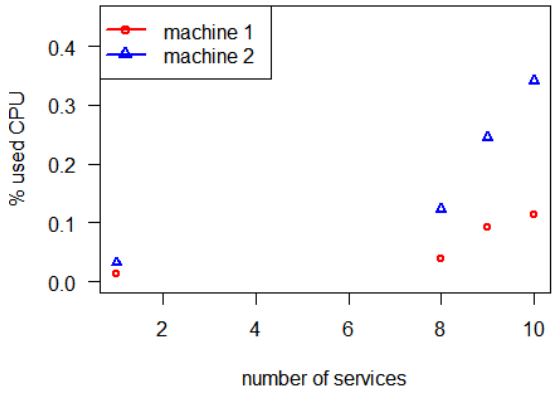

Figure 6.2: Example of measurement results for CPU utilization which cannot be approximated by sigmoid functions. Sample results from Google data traces from 2011 [95]

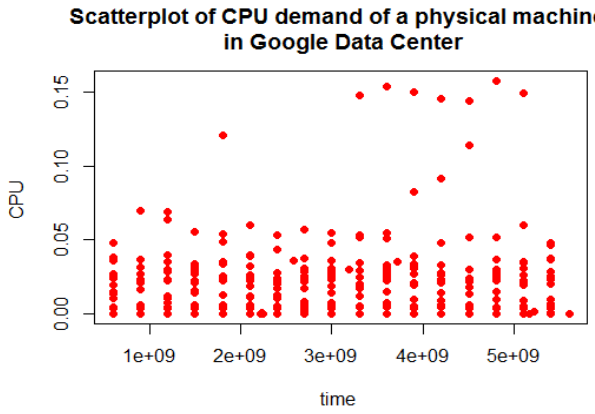

Figure 6.3: Workload profile of a PM from Google traces [95].

6.3.2 Allocation Algorithms

There are various algorithms for cloud services' allocation on servers, such as branch-and-bound (B&B), first fit (FF), first fit decreasing (FFD), best fit decreasing (BFD) [139], Linear-Program-relaxation-based heuristic (LP-relaxation) [8], Polynomial Time Approximation Schemes (PTAS) described by Chekuri and Khanna [138], hybrid algorithms with two phases where the former phase uses LP-relaxation and the second phase uses integer programming [8], etc. These different algorithms can obtain good results with computation time dependent upon the number of PMs. Martin Bichler et al. [8] assume that infeasible solutions are found more quickly than feasible solutions and they use an iterative approach for hybrid algorithms; they introduce lower bound (LB) for the number of servers, and when no solution is found with this LB number of servers, they increase the LB value by 1 until a solution is reached. In our heuristics, we include this iterative approach with incremental lower bound ($|K| = LB$) for the number of servers; thus, we have at least $|K|!$ concurrent solutions, which can only be resolved with difficulty by B&B algorithms [153]; therefore, to reduce the computational time of feasible solution, we add different small cost amounts for servers with same costs as suggested in [8]. Hua-Jun Hong et al. [126] use a polynomial algorithm to solve their quality of experience (QoE) problem and they calculate QoE degradation for each end-user as described in [122] [123]; then, they sort servers on QoE degradation. For our heuristics, we sort servers according to their penalties on SLAs or their net profits (see Figure 6.10).

6.3.3 Data Preprocessing

Martin Bichler et al. [8] highlight that the data preprocessing method is an important step in server consolidation. This method shows periodic patterns in workload traces; these patterns help to derive estimate parameters for client resource demands which vary over time. These resource periodic patterns can show if it is possible to formulate these demands by sigmoid or other regression models; thus, parameters of these estimations can be derived either offline from regressions of homogeneous servers or online from regressions of heterogeneous physical machines [126]. Google traces from 2011 [95], used as a base to test our algorithms, can not be regressed by sigmoid functions as shown in Figure 6.6.

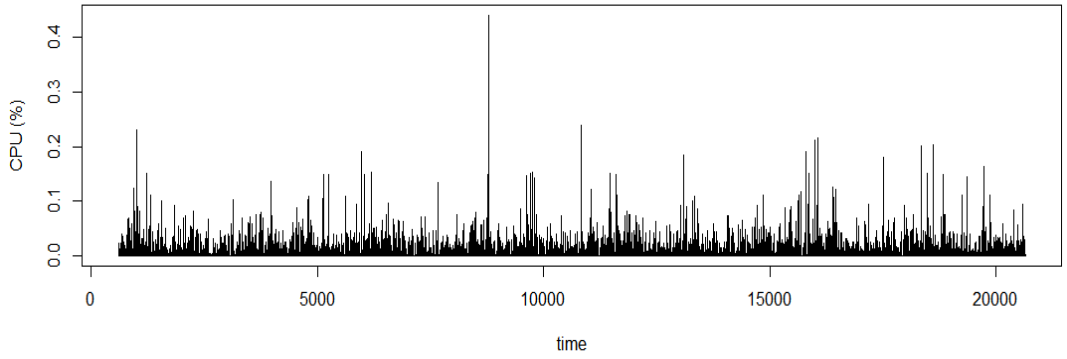

Figure 6.4: Workload profile of a sample Google [95].

machine_ID	task_index	CPU_rate
Min. :5.000e+00	Min. : 0	Min. :0.000
1st Qu.:7.777e+06	1st Qu.: 81	1st Qu.:0.000
Median :3.293e+08	Median : 385	Median :0.001
Mean :1.300e+09	Mean : 1967	Mean :0.017
3rd Qu.:2.275e+09	3rd Qu.: 1590	3rd Qu.:0.020
Max. :6.248e+09	Max. :20009	Max. :0.541

Figure 6.5: Summary of a sample of Google data traces [95].

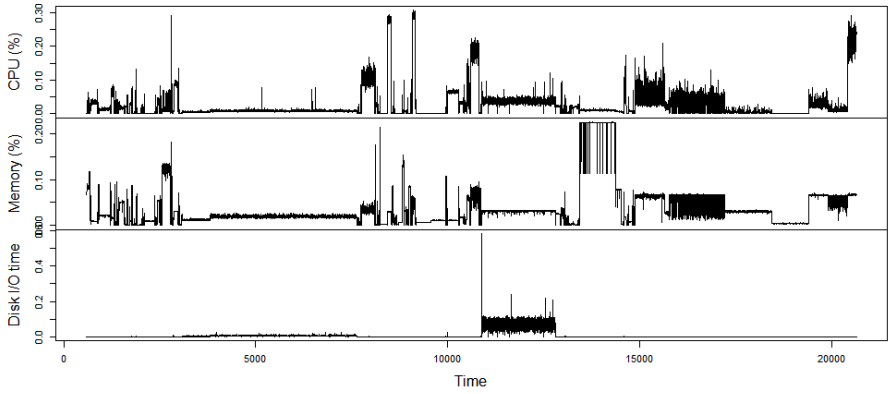

Figure 6.6: Example of Google physical machine' s usage timeline [95].

The distributions of CPU and memory are also presented in Figure 6.7 and Figure 6.8. These figures show the frequent value of resources' utilization in Google traces of 2011 [95].

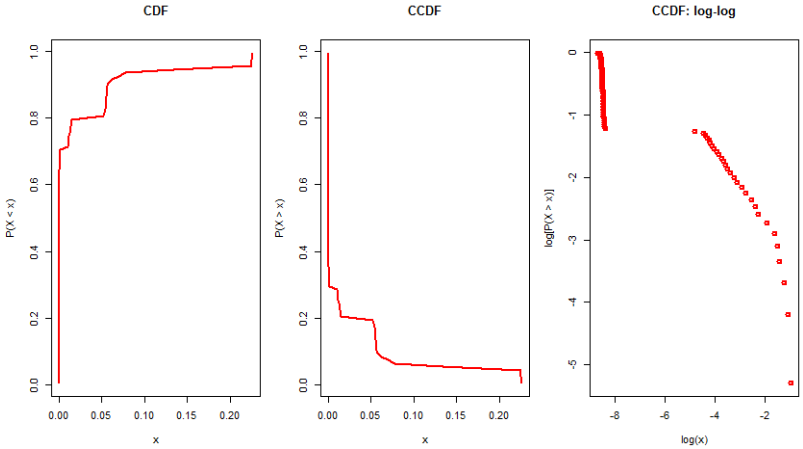

Figure 6.7: Distribution of memory usage of tasks of Google traces [95].

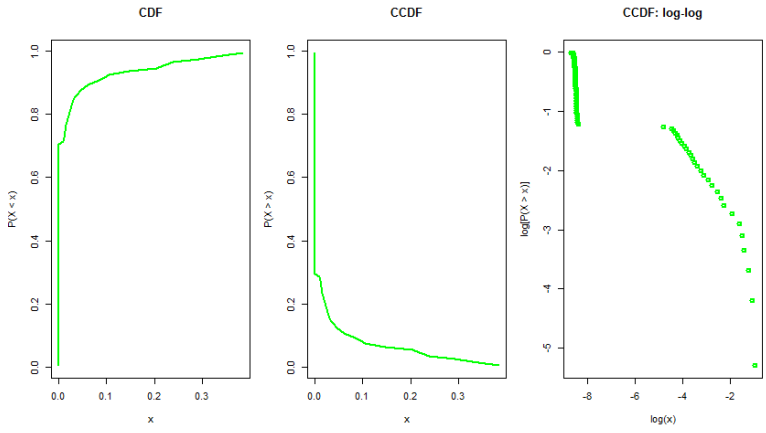

Figure 6.8: Distribution of CPU usage of tasks of Google traces [95].

As noticed in [154], some studies aim to predict data traces in a data center. They may use some techniques to success their predictions; for example, they may start by manipulating the raw data into an acceptable format for modeling and they may aggregate them to generate modeling data sets. Raw data manipulation provided for resource traces from two previous years may help to predict traces for the subsequent year. The manipulated data may be used as purely data driven. As predictive modeling, there are multiple models that can built on manipulated data to generate candidate solutions with the objective function to minimize the following formula [154]:

$$\sqrt{\frac{1}{n}\sum_{i}^{n}[(pred_{1i})-(act_{1i})]^2}$$

Where n is the members' count, $pred_i$ is the predicted value of a member i, and act_i is its actual value.

According to [154], this formula is the root of the mean squared error, or RMSE; fortunately, there are many algorithms in the literature to minimize this RMSE. Some of these algorithms are already freely available in the R language for statistical computing; for example, Gradient Boosting Machines [155], Neural Networks Back Propagation [156], Bagged Trees Random Forests [157], and Linear Models [158] are algorithms which can find individual candidate solutions. These algorithms combined may produce solutions better than their individual solutions because of their synergy to find solutions by different paths [154].

6.3.4 Simulation Design and Dependent Variables

Our simulations are designed to find out which characteristics have more impacts on our problem solutions in terms of problem size, solution quality, etc. For this reason, we study many characteristics such as service type, server resource capacities, and heuristics parameters. These characteristics are tested with various values to derive acceptable parameters which give good solutions in terms of profits, penalties, running times, number of required physical machines, etc. Our solution analysis has two main components; the former is the quality defined by the number of required servers and their costs whereas the second component is the computational time to find solutions. To find solutions, we use CPLEX to get results with B&B algorithms, and our own developed C# heuristics to get the results described in this paper in addition to other algorithms

such as FF and FFD. Our simulations use Windows on a PC with Intel(R) Core I7 4 GHz and 16 GB memory.

6.4 Formulation and Solution of VM Live Placement Problem

In this section, we present the system overview; then, we develop our mathematical model and algorithm for the penalty-centric optimization problem.

6.4.1 System Overview

Our system architecture is composed of K physical servers, I virtual machines, M tasks, U end-users, and a broker. Many virtual machines may execute many tasks on a physical machine. Physical servers, located in different data centers, serve clients who request various services via Internet from mobile devices, laptops, and desktops.

Our software monitors resources and implements a live placement heuristic; it monitors the workload of servers and executes live migration of virtual machines in order to achieve a successful compromise between the overall cloud service penalties and operational costs of the resources. Generally, the profit is more important for cloud services providers while the quality of service induced by minimal services' penalty is more critical for end-users. We study the second case in this section while the first case is presented in the following section. Cloud services may need different resource requirements, such as memory and CPU [159], and heterogeneous network resources such as bandwidth and latency. Moreover, end-users have a certain tolerate bounds for penalty levels for each different type of cloud services [160].

6.4.2 Notations and Models

In this paper, we study the virtual machines live placement problem, in which the live virtual machine placement decisions have a critical incidence on their interdependent service quality, operational costs, CPU delays, and network latencies. Table 6.1 and Table 6.2 present various symbols used by this paper's formula. Our parameters are obtained using the approach described in Figure 6.9.

Question ➔ input data ➔ characteristics ➔ parameters

➔ algorithm ➔ evaluation ➔ validation.

Figure 6.9: General strategy to get regression of input data.

Also, we use $Q\_ResUsedByTask_{m,n}$ as the quantity of a resource n used by a task m (such as CPU, etc.); Hua-Jun Hong et al. [126] indicate that sigmoid functions can model $Q\_ResUsedByTask_{m,n}$ correctly for some application with:

$$Q\_ResUsedByTask_{m,n} = \delta_{m,n,1} /(1 + e^{-\delta_{m,n,2}})$$

where $\delta_{m,n,1}$ and $\delta_{m,n,2}$ are obtained by regression.

Table 6.1: Parameters used by this paper's formulas

Term	Description
$PM\_downBound_{k,n}$	: Down bound of resource n for PM k.
$PM\_upBound_{k,n}$	: Upper bound of resource n for PM k.
$Max\_T\_dependency\_respTime_m$	: Maximal delay accepted to transmit shared data between a task m and its interdependent tasks.
$Max\_Res\_migr\_preCopy_{k,n}$	: Maximal quantity of resource n of PM k used to pre-migrate its VMs.
$Max\_Res\_migr\_finalCopy_{k,n}$	: Maximal quantity of resource n of PM k used to final-migrate its VMs.
$Max\_T\_preCopy\_VM_i$	: Maximal accepted total delays for pre-copy migration of VM i from a PM to another.
PMs_Count	: Total number of physical machines.

Term	Description
VMs_Count: Total number of virtual machines.	
$Task\_on\_VM_{i,m}$: Binary value equal to one if task m is hosted by VM i; otherwise, it is equal to zero.	
$SLA\_latency\_respTime\_task_m$: Service level contract on the response time of a task m hosted by a VM.	
$Interdependent\_Tasks_{m1,m2}$: Binary value equal to one if task m_1 and m_2 are interdependent; otherwise, it is equal to zero.	
$Max\_Res\_migr\_preCopy_{k,n}$: Maximal accepted quantity of resource n to be used exclusively by a physical machine k when it executes a pre-copy migration of one of its VMs.	
$Max\_Res\_migr\_finalCopy_{k,n}$: Maximal accepted quantity of resource n to be used exclusively by a physical machine k when it executes a final-copy migration of one of its VMs.	
$TransInterMemory_{m_1,m_2}$: Quantity in GB of memory to transfer from a task m_1 to task m_2	
K_t: Transmission rate of a task.	
S_m: Propagation speed of a link connecting physical machines hosting VMs which host task m_1 and task m_2.	
K_r: Reception rate of a task.	
$Max\_P\_task_m$: Maximal accepted penalty on delays of a task m.	
g_m: Profit obtained by cloud provider with a service m.	

Term	Description
$c_{k,n}$: Unit operational cost of a resource n on a physical machine k.	
w_n : Operational cost of a resource n of a physical machine k.	
d_{k_1,k_2}: Physical distance between PM k_1 and PM k_2.	
risk_percentage: Pourcentage of confidence interval percentile which corresponds to decision maker's risk attitude (95% by default).	
interval_delay: Interval delay corresponding to aggregating period of data observation (24 hours by default).	
Simulation_timeout: timeout for our tested algorithms (30 minutes as default).	

According to [123] and [126], response time $T\_latency\_respTime\_task_m$ of user requests has a great impact upon QoS perception, and it is an aggregation delay of the end-user interaction, network, CPU processing, and interdependency delays of a service m; therefore, the $T\_latency\_respTime\_task_m$ formula can omit the end-user interaction delay as it is independent of VM live placement. Also, we write the penalty of a service P_task_m as the result of division of the total delay $T\_latency\_respTime\_task_m$ of a task m by its service level agreement contract $SLA\_latency\_respTime\_task_m$.

Table 6.2: Variables used by this paper's formulas

Term	Description
P_task_m : Penalty of a task m.	
$TasksByVM_i$: Number of tasks on a virtual machine i.	
$x_{k,i}$: Decision binary variable equal to one if VM i is hosted by PM k; otherwise, it is equal to zero.	

Term	Description
$UsedResByVM_{i,n}$: Quantity used of a resource n by VM i.	
$prev\_x_{k,i}$: Binary variable equal to the binary decision variable $x_{k,i}$ found in the previous round of placement of virtual machines.	
$T\_migr\_VM_i$: Migration delay of VM i from a PM to another. By default, it is equal to zero when this VM i is not migrated.	
$T\_dependency\_respTime_m$: Total of transmission delays between task m and its interdependent tasks.	
$Q\_Res\_migr\_preCopy\_VM_{i,n}$: Quantity of the resource n used exclusively for the pre-copy migration of VM i. If this VM i is not migrated, this resource n quantity is equal to zero.	
$Q\_Res\_migr\_finalCopy\_VM_{i,n}$: Quantity of the resource n used exclusively for the final-copy migration of VM i. If this VM i is not migrated, this resource n quantity is equal to zero	
$T\_preCopy\_VM_i$: Total pre-copy migration delays of VM i from a physical machine to another.	
$T\_latency\_respTime\_task_m$: Response time (latency) of a task m hosted on a virtual machine.	
$T\_dependency\_downTransfert\_task_{m_1,m_2}$: Delay to transmit shared data of a task m_1 to the network from its hosting virtual machine towards its interdependent tasks hosted by other virtual machines.	
$T\_dependency\_propagation\_task_{m_1,m_2}$: Delay to propagate shared data of a task m_1 between its hosting virtual machine and its interdependent tasks hosted by other virtual machines.	

Term	Description
$T\_dependency\_upTransfert\_task_{m_1,m_2}$: Delay to get shared data of a task m_1 from the network delivered by its hosting virtual machine towards its interdependent tasks hosted by other vms.	
$T\_downTransfert\_VM_i$: Total delay to transmit a virtual machine i to the network from its hosting physical machine towards another pm.	
$T\_propagation\_VM_i$: Total delay to propagate a virtual machine i between two physical machines.	
$T\_upTransfert\_VM_i$: Total delay to get a virtual machine i from the network sent by its initial hosting physical machine towards its destination physical machine.	
$Q\_maxLevel\_Res_n$: Maximal value of each resource of virtual machines.	
$Task\_on\_PM_{k,m}$: Binary value equal to one if task m is hosted by PM k; otherwise, it is equal to zero.	
$x_{k,n}$: Binary variable equal to one if resource n is used by PM k; otherwise, it is equal to zero.	
$UsedResByPM_{k,n}$: Quantity used of a resource n by PM k.	
Q_Avg_UseRes$_n$: Average use of resource n per a period.	
$Q\_ResUseByTask_{m,n}$: Utilization quantity of a resource n (such as CPU) by a task m.	
d_k : Processing delay to serve an end-user by a physical server k hosting services.	
e_k : Round-trip delay between a physical server k and users.	
g_i : Profit obtained by cloud provider with a virtual machine i.	

Term	Description
g_k: Profit obtained by cloud provider with a physical machine k hosting virtual machines.	
w_k : Total operational cost of all resources of a physical machine k.	
d_{m_1,m_2}: Physical distance between PMs hosting VMs which host task m_1 and task m_2.	

6.4.3 Problem Formulation

In our model, we note $x_{k,i} \in \{0,1\}$ $(1 \leq i \leq I,\ 1 \leq k \leq K)$ as decision variables, where $x_{k,i} = 1$ if and only if virtual machine i is hosted on physical machine k. We present the penalty-centric problem with formulas shown below:

$$Min \left[\sum_{m=1}^{M} P_{task_m} \right] \tag{6.1}$$

$$\text{s.t.} \quad TasksByVM_i \geq 1 ==> \sum_{k=1}^{K} x_{k,i} = 1 \qquad \forall i \tag{6.2}$$

$$TasksByVM_i \geq 1 \ \ and \ \ x_{k,i} = 1 ==> \qquad \forall k, \forall i, \forall n \tag{6.3}$$

$$PM_{downBound_{k,n}} \leq \sum_{ii=1}^{I} x_{k,ii} * UsedResByVM_{ii,n} \leq PM_{upBound_{k,n}}$$

$$\left(x_{k_1,i} = 0 \ \ and \ prev_{x_{k_2,i}} = 1\right) \text{ and } \left(x_{k_2,i} = 1 \ and \ prev_{x_{k_2,i}} = 0\right) \qquad \forall i, \forall k_1, \forall k_2 \tag{6.4}$$

$$==> T_{propagation_{VM_i}} = \frac{d_{k_1,k_2}}{s_m}$$

$$x_{k,i} = 1 \ and \ prev_{x_{k,i}} = 0 ==> \quad \forall i, \forall k \qquad (6.5)$$

$$T_{migr_{VM\,i}} = T_{downTransfert_{VM\,i}} + T_{propagation_{VM\,i}} + T_{upTransfert_{VM\,i}}$$

$$T\_dependency\_respTime_m \leq Max\_T\_dependency\_respTime_m \qquad \forall m \qquad (6.6)$$

$$x_{k,i} * Q\_Res\_migr\_preCopy\_VM_{i,n} \leq Max\_Res\_migr\_preCopy_{k,n} \qquad \forall k, \forall i, \forall n \qquad (6.7)$$

$$x_{k,i} * Q\_Res\_migr\_finalCopy\_VM_{i,n} \leq Max\_Res\_migr\_finalCopy_{k,n} \qquad \forall k, \forall i, \forall n \qquad (6.8)$$

$$x_{k,i} \in \{0,1\} \quad \forall 1 \leq i \leq I, \forall 1 \leq k \leq K \qquad (6.9)$$

1 : **initialize** of all previous VMs placements $prev\_x_{k,i}$ to zero
2 : **for** each round of VM live placement $r = 1,2, \dots, R$ **do**
3 : **let** solutionFound = false
4 : **while** solutionFound is false **do**
5 : **solve** VMs placement problem with a simplex algorithm (SLV).
6 : **if** a solution, satisfying constraints in Eqs. (6.2) — (6.9), is found for VMs placement problem **then**
7 : **let** $Q\_maxLevel\_Res_n$ = maximal value of a resource n used by virtual machines in the recent round of VM live placement.
8 : **let** $prev\_x_{k,i}$ = actual $x_{k,i}$ of the recent round of VM live placement.
9 : **let** solutionFound = true
10 : **else if** placement solution is not found **then**
11 : **add** a new physical machine i.e. LB = LB + 1.
12 : **set** the upper bounds of each resource of the new physical machine to the maximal level $Q\_maxLevel\_Res_n$ increased by a parameterized coefficient$Coef_maxLevel_Res$.
13 : **initialize** all previous VMs placement $prev\_x_{k,i}$ to zero.
14 : **end if**
15 : **end while**
16 : **end for**

Figure 6.10: Pseudo code of the first version of our algorithm PCH.

The objective function in Eq. (6.1) minimizes the total penalties on response time of all services hosted by virtual machines hosted themselves by physical machines, i.e. the sum of the division of response times by their service level agreement (SLA). Eq. (6.2) ensures that each virtual machine, hosting one or many tasks, is hosted only by one physical machine. Eq. (6.3) makes sure that the upper and lower bounds of every resource of a PM are respected by the total of the resource quantities of all the tasks hosted by this PM. Eqs. (6.4) and (6.5) derive the propagation and migration delays of a virtual machine as intermediate variables. Eq. (6.6) impose delay constraint between interdependent VMs. Eqs. (6.7) and (6.8) impose resources constraints on each virtual machine. As described above, the formulation minimizes the penalties on response time while respecting the limits of resource utilization on each virtual machine and physical machine. Variables used in the Eqs. (6.1) — (6.9) are calculated as below:

$$P\_task_m = \frac{T\_latency\_respTime\_task_m}{SLA\_latency\_respTime\_task_m} \qquad \forall m \qquad (6.10)$$

$$TasksByVM_i = \sum_{m=1}^{M} Task\_on\_VM_{i,m} \qquad \forall i \qquad (6.11)$$

$$UsedResByVM_{i,n} = \sum_{m=1}^{M} (Task\_on\_VM_{i,m} * \ Q\_ResUseByTask_{m,n}) \qquad \forall i, \forall n \qquad (6.12)$$

$$T\_dependency\_respTime_m = \qquad \forall m \qquad (6.13)$$

$$\sum_{m_2=1}^{M} Interdependent\_Tasks_{\mathrm{m},m_2}$$

$$* \ (T\_dependency\_downTransfert\_task_{m,m_2}$$

$$+ \ T\_propagation\_task_{\mathrm{m},m_2}$$

$$+ \ T\_dependency\_upTransfert\_task_{\mathrm{m},m_2})$$

$$T\_preCopy\_VM_i = \quad \forall i \qquad (6.14)$$
$$T\_downTransfert\_VM_i + T\_propagation\_VM_i + T\_upTransfert\_VM_i$$

$$T\_latency\_respTime\_task_m = \quad \forall m \qquad (6.15)$$
$$\left(\sum_{k=1}^{K} Task\_on\_PM_{k,m} * d_k\right) + T\_dependency\_respTime_m + \left(\sum_{i=1}^{I} T\_downTransfert\_VM_i * Task\_on\_VM_{i,m}\right) + \left(\sum_{k=1}^{K} e_k * Task\_on\_PM_{k,m}\right)$$

$$T\_dependency\_downTransfert\_task_{m_1,m_2} = \frac{TransInterMemory_{m_1,m_2}}{K_t} \qquad \forall m_1, \forall m_2 \quad (6.16)$$

$$T\_dependency\_propagation\_task_{m_1,m_2} = \frac{d_{m_1,m_2}}{s_m} \qquad \forall m_1, \forall m_2 \quad (6.17)$$

$$T\_dependency\_upTransfert\_task_{m_1,m_2} = \frac{TransInterMemory_{m_1,m_2}}{K_r} \qquad \forall m_1, \forall m_2 \quad (6.18)$$

$$T\_downTransfert\_VM_i = \frac{UsedResByVM_{i,MEMORY}}{K_t} \qquad \forall i \qquad (6.19)$$

$$T\_upTransfert\_VM_i = \frac{UsedResByVM_{i,MEMORY}}{K_r} \qquad \forall i \qquad (6.20)$$

$$Q\_maxLevel\_Res_n = Max_{m \in M}(Q\_ResUseByTask_{m,n}) \qquad \forall n \qquad (6.21)$$

$$x_{k,n} = (PM\_upBound_{k,n} > 0) \qquad \forall k, \forall n \qquad (6.22)$$

$$UsedResByPM_{k,n} = \sum_{i=1}^{|I|} (x_{k,i} * UsedResByVM_{i,n}) \qquad \forall k, \forall n \qquad (6.23)$$

$$nbrTasksByPM_k = \sum_{m=1}^{M} x_{k,i} * TasksByVM_i \qquad \forall k \qquad (6.24)$$

$$g_i = \sum_{m=1}^{M} g_m * Task\_on\_VM_{i,m} \qquad \forall i \qquad (6.25)$$

$$g_k = \sum_{i=1}^{I} (x_{k,i} * g_i) \qquad \forall k \qquad (6.26)$$

$$w_k = \sum_{n=1}^{N} (x_{k,n} * w_n * UsedResByPM_{k,n}) \qquad \forall k \qquad (6.27)$$

$$Task\_on\_PM_{k,m} = x_{k,i} * Task\_on\_VM_{i,m} \qquad \forall k, \forall m \qquad (6.28)$$

$$d_{m_1,m_2} = \sum_{\substack{k_1=1,\\ k_2=1}}^{K} (d_{k_1,k_2} * Task\_on\_PM_{k_1,m_1} * Task\_on\_PM_{k_2,m_2}) \qquad \forall m_1, \forall m_2 \qquad (6.29)$$

$$T\_finalCopy\_VM_i = \frac{Q\_Res\_migr\_finalCopy\_VM_{i,MEMORY}}{K_t} \qquad \forall i \qquad (6.30)$$

$$nbrTasks = \sum_{k=1}^{K} nbrTasksByPM_k \qquad (6.31)$$

6.4.4 Proposed Algorithm

Formulation in Eqs. (6.1) — (6.9), of the penalty-centric-live-placement-problem, can be exactly solved for small–sized instances by an optimization solver such as CPLEX [33]. These solver-based algorithms, referred to in this paper as SLV, find optimal solutions with exponential computation complexity; for this reason, we propose an efficient approximation heuristic, referred to as penalty-centric-heuristic (PCH), and we test its performance on google traces from 2011 [95]. The PCH algorithm (see Figure 6.10) looks for good VM consolidations without PM resource capacity violations. For each round of VM live placement, there are two steps: in the former, the algorithm PCH first tries to find a solution with SLV constraints described in Eqs. (6.2) — (6.9); then, if a solution is not found, the PCH algorithm executes the second step, where it adds a new PM with appropriate bounds to be reused in the former step. This version of PCH runs in polynomial time with the exception of the SLV part. For this reason, we develop a second version of PCH, based on tabu meta-heuristic (see Figure 6.11), which runs in polynomial time.

6.5 Alternative Formulation and Algorithms

This section proposed an alternative formulation and algorithm for the VM live placement problem introduced in Section 6.4 as the penalty-centric problem (more suitable for closed cloud services). This section, contrary to the penalty-centric approach, proposes a provider-centric approach suitable for open cloud services where maximizing the overall revenue is more relevant for cloud providers. We use g_m as the cloud provider gain periodically paid by an end-user to use a service m. We denote the total operational cost of all used resources on a physical machine k such as memory and CPU as:

$$w_k = \sum_{n=1}^{N} c_{k,n} * u_{k,n} \qquad (6.32)$$

The term $c_{k,n}$ indicates the individual operational cost of a resource n on a physical machine k including many costs such as maintenance, energy, and depreciation. Last, we consider servers sending data at K_t bps and receiving data at K_r bps.

Next, we add the following new constraint (6.33) to take into consideration service penalties:

$$P\_task_m \leq Max\_P\_task_m \qquad \forall m \qquad (6.33)$$

Therefore, our new provider-centric formulation is derived from $Eqs.(6.2) - (6.9)$, presented in Section 6.4, where the objective function in $Eq.(6.1)$ is replaced with the following $Eq.(6.34)$, thus maximizing the total revenue:

$$\max[\ \sum_{k=1}^{K} (g_k - w_k)\ ] \qquad (6.34)$$

This new derived model, referred as SLV', is more suitable for provider-centric concerns and it can be solved by a solver-based algorithm suitable for small-sized instances. Then, we develop an alternative heuristic PCH' derived from PCH and where are modified from $Eqs.(6.2) - (6.9)$ to $Eqs.(6.2) - (6.9)$ and (6.33). This PCH' looks in each round for a solution respecting constraints in $Eq.(6.33)$ and $Eqs.(6.2) - (6.9)$. This version of PCH' runs in polynomial time except for the SLV' part. For this reason, we develop a second version of PCH', based on tabu meta-heuristic, which runs in polynomial time.

6.6 Heuristics

This paper penalty-centric live placement problem aims to minimize the objective function in $Eq.(6.1)$ subject to constraints in $Eqs.(6.2) - (6.9)$. This problem is NP-hard, difficult to solve with enumerative searches and other standard methods unsuitable for large-sized instances; for this reason, we replace the SLV part of the PCH algorithm by an approximation tabu search (see Figure 6.11).

- Step 1 (Getting an initial solution) :
  - ✓ Find, as follows, an initial solution for the penalty-centric VM live placement problem respecting constraints in Eqs. (6.2) — (6.9) :
    - ➢ Use FFD algorithm [139] to assign virtual machines to physical machines.
- Repeat the following steps 2 and 3 for "maxIterations" times:
  - ✓ Step 2 : Repeat the following steps (2.1 and 2.2) for "maxNeighbours" times:
    - ➢ Step 2.1 (Getting best solution from neighborhood) :
      - ❖ 2.1.1. Find the best move to obtain a solution, respecting constraints in Eqs. (6.2) — (6.9), using possible tabu moves and aspiration criteria.
      - ❖ 2.1.2. Define a number of iterations for which the chosen move is considered tabu (this number is chosen from a uniform distribution).
    - ➢ Step 2.2 (Updating the best TS of best solutions): If the penalty of the current solution is less than the penalty of the best solution already found, the current solution becomes the best solution.
  - ✓ Step 3 (Updating the best of best solutions) :
    - ➢ If the penalty of the current solution is less than the cost of the best solution already found, the current best solution becomes the best of the best solutions.

- Step 1 (Getting an initial solution) :
  - ✓ Find, as follows, an initial solution for the penalty-centric VM live placement problem respecting constraints in Eqs. (6.2) — (6.9) :
    - ➢ Use FFD algorithm [139] to assign virtual machines to physical machines.
- Repeat the following steps 2 and 3 for "maxIterations" times:
  - ✓ Step 2 : Repeat the following steps (2.1 and 2.2) for "maxNeighbours" times:
    - ➢ Step 2.1 (Getting best solution from neighborhood) :
      - ❖ 2.1.1. Find the best move to obtain a solution, respecting constraints in Eqs. (6.2) — (6.9), using possible tabu moves and aspiration criteria.
      - ❖ 2.1.2. Define a number of iterations for which the chosen move is considered tabu (this number is chosen from a uniform distribution).
    - ➢ Step 2.2 (Updating the best TS of best solutions): If the penalty of the current solution is less than the penalty of the best solution already found, the current solution becomes the best solution.
  - ✓ Step 3 (Updating the best of best solutions) :
    - ➢ If the penalty of the current solution is less than the cost of the best solution already found, the current best solution becomes the best of the best solutions.

Figure 6.11: Algorithm of our TS (approximating SLV) where "maxIterations" and "maxNeighbours" are input parameters.

In this paper, we use a tabu search (TS) which is an adaptive technique used generally to solve NP-hard problems for large-sized instances [109]. Our TS, based on [161], looks for good solutions that approximate the optimal solutions of the penalty-centric problem. Our TS has three steps (see Figure 6.11); the first finds an initial solution that respects all of the problem constraints described

in $Eqs.(6.2)-(6.9)$; then, the TS executes many iterations of the second and third steps; in the second step, the TS looks outside a tabu list [109] [162] [110] for better solutions in the neighborhood of a solution already found; the last step updates the best of the best solutions by comparing it with the best solution already found in the second step.

The neighborhood of a current solution, used in the second step of our TS, is a set of solutions obtained by applying some moves (transformations) on a current solution; our TS move consists of moving some virtual machines between physical machines; then, this move is put in a tabu list for a while; however, our TS uses the aspiration criterion technique, retiring a move from the tabu list when it helps to find a better solution. Lastly, our TS considers some solutions "tabu" and accepts some degradation of our objective function (see $Eq.(6.1)$) to avoid local minimum solutions. Our alternative TS' is derived from TS (see Figure 6.11) where constraints are modified from Eqs. $(6.2)-(6.9)$ to $(6.2)-(6.9)$ and (6.33) and the penalty calculus is modified to calculus of net profits. This TS' is used as approximation algorithm for SLV'. This TS' aims to maximize the overall gain of a cloud service provider as defined in $Eq.(6.34)$ subject to constraints in $Eqs.(6.2)-(6.9)$ and (6.33).

6.7 Trace-Driven Simulation Results

This section describes our simulation results for large-sized instances based on Google traces [95].

6.7.1 Setup

Our simulator is designed with C# (framework .Net 4.5) to test our hybrid algorithm (PCH/PCH') on Google traces [95]. We also computed Google machines' count and performances to construct our benchmark. We use Google traces from 2011 [95] in order to test the efficiency of our algorithm. These Google traces contain also descriptions of Google physical machines and their tasks in 2011; Figure 6.5 shows a summary of a sample of Google trace file. These Google trace samples from 2011 are nearly 39 GB for 29 days and are decomposed in 20,119 files. In our simulations, we both use all of this data and randomly choose samples of this data for training, testing and validating our algorithm. As some traces do not have complete information, we ignore missed data and we consider only data traces having significant information.

We use Google physical machines with different capacities for each of their resources ranging from $PM_Res_Capacity_$inf to $PM_Res_Capacity_$max (see Table 6.3); our simulations use K physical machines where K can be $K_1, K_2, K_3, K_4, or\ K_5$ (see Table 6.3). Also, we use many virtual machines and tasks with different resource capacities from $Task\_Res\_Capacity\_\text{inf}_n$ to $Task\_Res\_Capacity\_\text{max}_n$. For our tests, we use three different resource types which are memory, CPU, and disk usage. For each algorithm, we execute simulations for $Simulation\_length$ periods and each period is exected for the $execution\_period\_duration$ (see Table 6.3) and we plot their average values (penalties, profits, running time, number of required physical machines, etc.) as shown in Figure 6.12 to Figure 6.18. These simulations are developed and executed on a PC with Intel(R) Core I7 4 GHz and 16 GB memory. The rest of this section use default values of Table 6.3.

6.7.2 Solution Penalties vs. Problem Size

We plot our algorithm results in terms of penalties in Figure 6.12; this figure shows that our PCH/PCH' algorithms outperform those of Google traces [95] up to limits cited in Table 6.3 which are $Gap\_penalty\_PCH\_GOG\_max$ and $Gap\_penalty\_PCH\_GOG\_max$. Therefore, Figure 6.13 shows that our PCH/PCH' algorithms use less PMs than those used by the Google traces [95]. Figure 6.14 shows the variation of penalties in function of the number of physical machines. The Figure 6.19 and Figure 6.20 also show that penalties are always lower than $a\ certain\ penalty\_max$, which is less than the maximal penalty of existing configuration described in google traces [95]. PCH/PCH' are on average better than the exiting configuration in Google traces by $Average\_penalty\_improvement\_PCH\_GOG$ (see Table 6.3).

Table 6.3: Simulation parameters and results

Term	Description and values
$K_1, K_2, K_3, K_4\ and\ K_5$	: Different values of initial count of physical machines. In this paper's simulation, they are by defaults equals respectively to 1%, 10 %, 50 %, 80 %, and 100 % of 12,583 PMs of Google traces [95].

Term	Description and values
$PM\_Res\_Capacity\_inf$	Minimal value of capacities of physical machines of our simulation Google traces from 2011 [95]. This value is set to "0.25" for CPU and "0.03" for Memory by Google traces (see Table 6.4) where CPU and memory are linearly scaled to obtain "1" as their maximum value.
$PM\_Res\_Capacity\_max$	Maximal value of capacities of physixal machine of our simulation Google traces [95]. This value is set to "1" for CPU and Memory by Google traces (see Table 6.4) where CPU and memory are scaled to obtain "1" as their maximum value.
$Task\_Res\_Capacity\_inf_n$	Minimal value of capacities of resource n of tasks of our simulation Google traces from 2011 [95]. This value is set to "0" for CPU and "0" for Memory by Google traces where CPU and memory are scaled to "1" as their maximum value.
$Task\_Res\_Capacity\_max_n$	Maximal value of capacities of resource n of tasks of our simulation Google traces from 2011 [95]. This value is set to "1" for CPU and "1" for Memory by Google traces where CPU and memory are scaled to "1" as their maximum value.
$Simulation\_length$	Number of periods treated by a simulation; by default, its value is equal to 500 which is the number of files of Google traces [95].
$Execution\_period\_duration$	Duration of each execution of one period of our algorithms; by default, its value is set to the duration obtained from a file of Google traces (in average one period is nearly 84 minutes).
$Risk\_percentage_i$	Different values of percentiles. In this paper's simulation, they are by defaults equals to 100%.
$Simulation\_interval\_delay$	The number of periods aggregated in each step of our simulations. By default, it is set to 1; however, it can take other values such as 5, 10, etc. which correspond to the number of subsequent files treated at once by a step of simulations.

Term	Description and values
$Simulation\_instances\_count$	: Number of instances used each time to test our algorithms on Google traces [95]. In our different simulations, we use 10 instances.
$Gap\_penalty\_PCH\_GOG\_max$	: Maximal improvement of penalty gap between PCH/PCH' and exiting configuration in Google traces [95]. We find it is around "99 % ".
$Gap\_profit\_PCH\_GOG\_max$	: Maximal profit gap between PCH/PCH' and exiting configuration in Google traces from 2011 [95]. In our different simulations, we find that it is around "100%".
$Average\_penalty\_improvement\_PCH\_GOG$	: Average of penalty improvement, with which PCH is better than the exiting configuration in Google traces from 2011 [95]. We find that it is around "13.50 %".
$RunningTime\_max$	: Maximal algorithms running time found using our simulation Google traces from 2011 [95]. In our different simulations, we find that it is around "169 minutes".
$Gap\_solvedInstance\_PCH\_GOG\_max$	: Maximal gap between percent of solved instances of PCH using our simulation with Google traces from 2011 [95]. In our different simulations, we find that it is around "1".
$Required\_physical\_machine\ \_max$	: Maximal number of solved required physical machines found using our simulation Google traces [95]. In our different simulations, we find that it is around "3214" which is only a third of the 12583 of installed PMs used on Google traces [95].

Table 6.4: Configuration of physical machines used by Google traces from 2011 [95]

# PMs	1	3	5	5	52	126	795	1001	3863	6732
CPUs	0.5	1	0.5	0.5	0.5	0.25	1	0.5	0.5	0.5
Memory	0.06	0.5	0.97	0.03	0.12	0.25	1	0.75	0.25	0.5

6.7.3 Solution Profits vs. Problem Size

We plot our algorithm results in terms of profits in Figure 6.14 and Figure 6.15; these figures show that our PCH/PCH' algorithms use less physical machines, and therefore, PCH/PCH' outperforms existing configuration described in google traces of 2011 [95] up to $Gap\_profit\_PCH\_GOG\_max$ (see Table 6.3); these performances are due to a decreased number of required PMs.

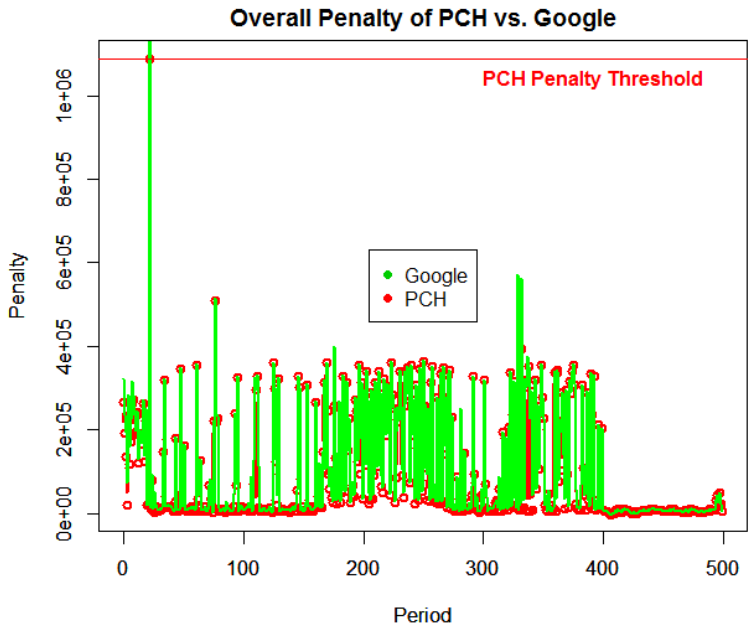

Figure 6.12: Penalty results are always lower than $Penalty\_max$ with PCH/PCH' applied on Google traces from 2011 [95].

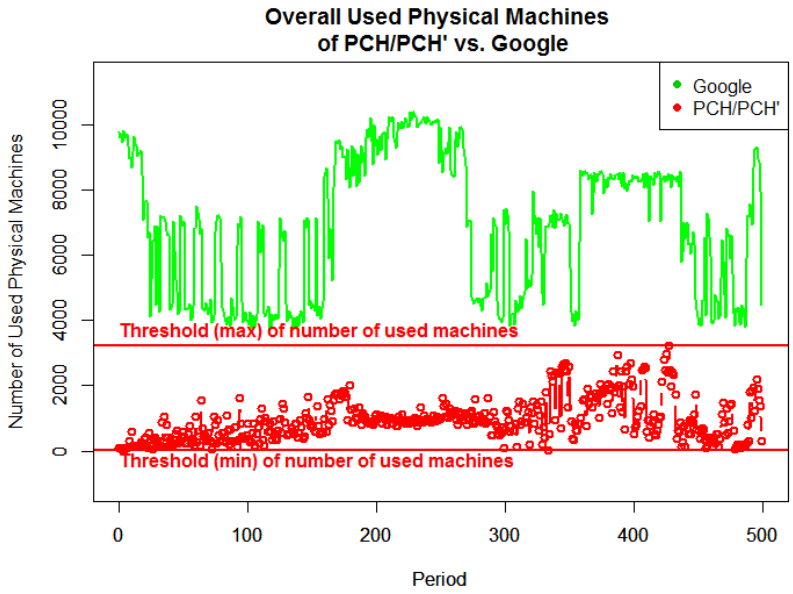

Figure 6.13: Number of required PMs are always lower than the threshold $Required\_physical\_machine\ \_max$ with Google traces from 2011 [95].

6.7.4 Running Time vs. Problem Size

Figure 6.16 plots the running time of different algorithms PCH/PCH' which are less than $RunningTime\_max$ (see Table 6.3); these average running times depend on the numbers of physical and virtual machines. Figure 6.17 shows the computation time of our PCH/PCH' algorithms for different samples; moreover, Figure 6.18 shows interesting gaps in solved instances with PCH/PCH' heuristics. These solved instances are executed in less than $RunningTime\_max$ and show interesting gaps near $Gap\_solvedInstance\_PCH\_GOG\_max$ (see Table 6.3). It is interesting to notice that some task types have more demands on resources than others and they have higher impacts on the gaps and percentage of solved solutions within $Execution\_period\_duration$; these tasks fit less in physical machines than others, and so, they increase the required physical machines and the computational delays.

Table 6.5 shows that running delays are relatively short even for K_5 physical machines within $3 * K_4$ resources (memory, CPU, and disk usage) and $4 * K_4$ virtual machines within many tasks (more than 20,000 tasks).

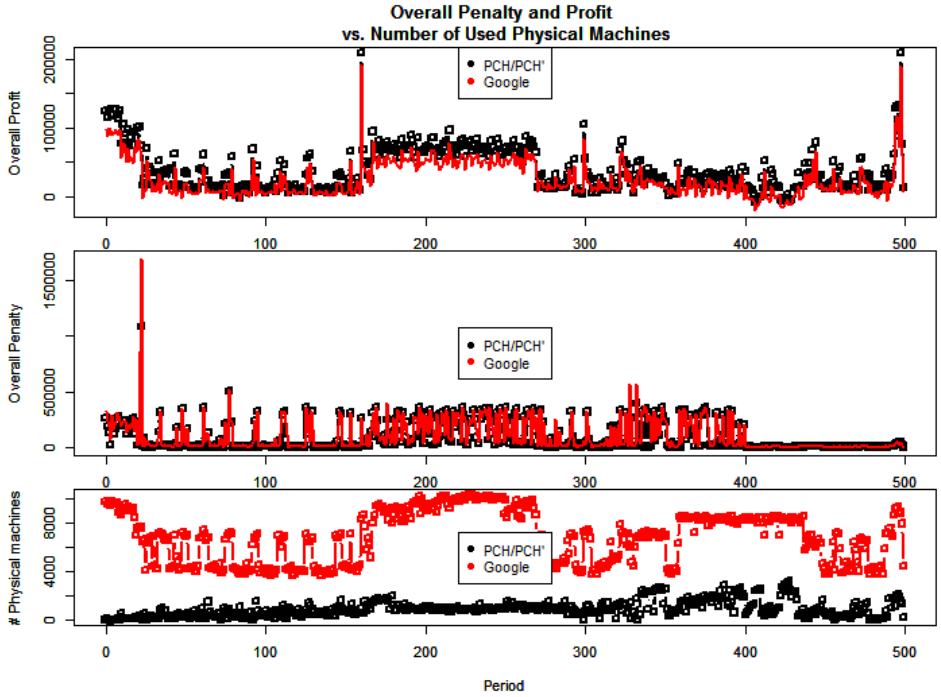

Figure 6.14: Impacts of number of required physical machines on penalties and net profits with Google traces from 2011 [95].

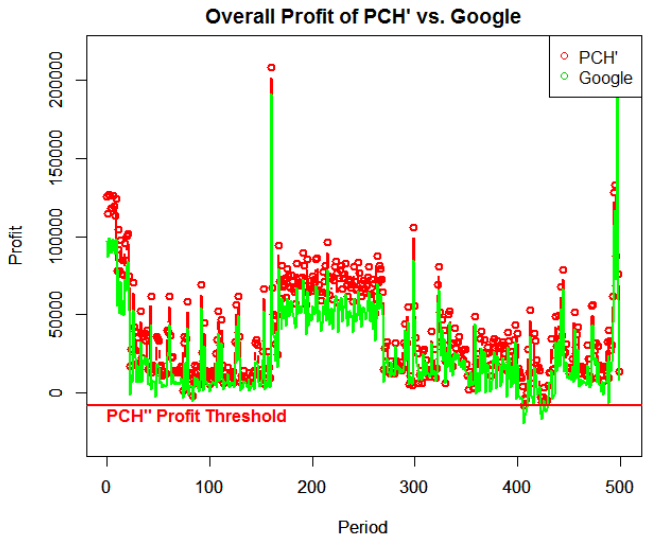

Figure 6.15: Profit results of PCH/PCH' is always higher than the existing Google configuration and above the threshold $Profit\_min$ with Google traces from 2011 [95].

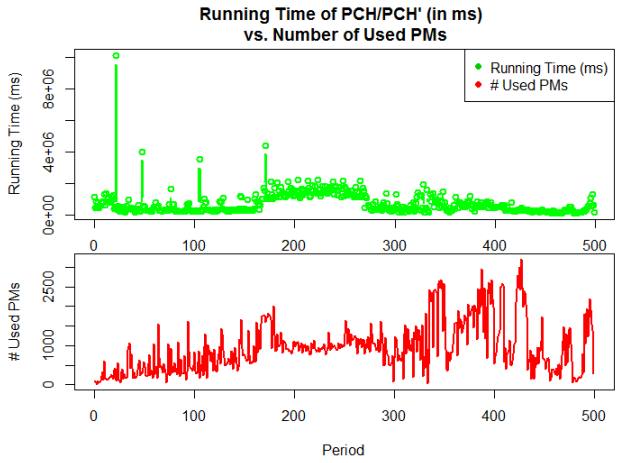

Figure 6.16: Running time of algorithms PCH/PCH' is lower than $runningTime\_max$ with Google traces from 2011 [95].

Table 6.5: Running time for PCH/PCH' with Google traces [95]

# PMs	Min (in sec)	Mean (in sec)	Max (in sec)
1% of 12583 PMs	23	33	213
10% of 12583 PMs	22	36	1075
50% of 12583 PMs	65	151	383
80% of 12583 PMs	131	398	1825
100% of 12583 PMs	126	733	10117

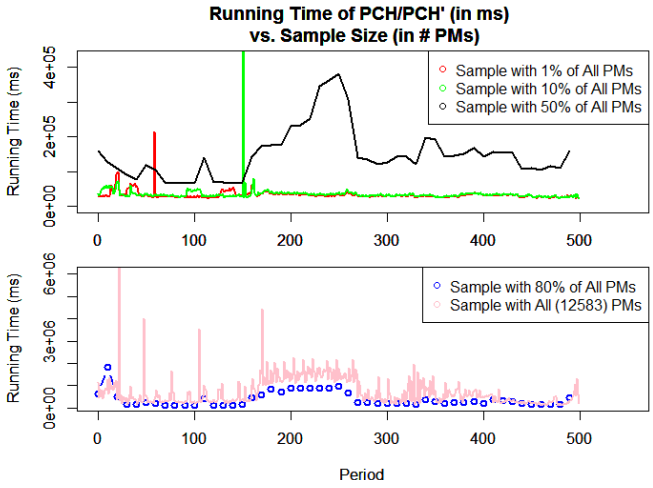

Figure 6.17: Running times for PCH/PCH' heuristics for different samples with Google traces from 2011 [95].

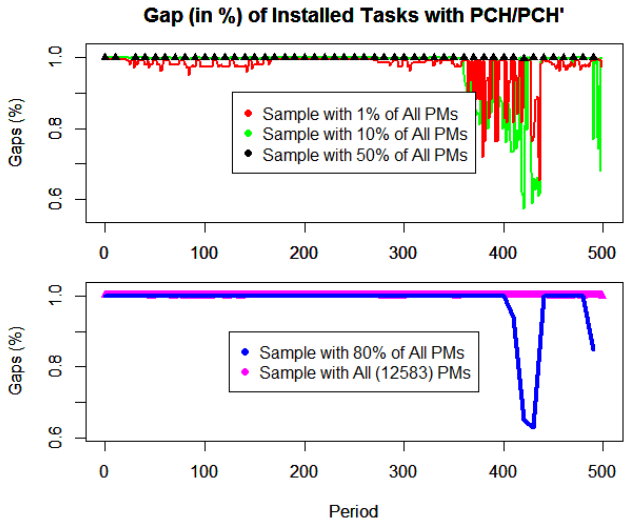

Figure 6.18: Proportion of solved instances (installed tasks) is practically equal to "1" with PCH/PCH' heuristics applied on Google traces from 2011 [95].

6.7.5 Summary

i. Live placement of virtual machines is an NP-hard optimization problem. As discussed in the introduction, live placement of virtual machines is NP-hard optimization problem as it is a variation of the NP-Complete virtual network embedding problem [117] solved approximately by [117] [118] [119] [120] [121].

ii. The solution time depends on the task types, their resource demands, and the server capacities. Some service types are less demanding for resources and can be grouped on the same physical machine whereas other service types are more demanding for resources and can be placed only with their corresponding negatively correlated types.

iii. Decreasing length in periods of a simulation step results in less required physical machines. Using small periods, to get their maximal resources' demands, gives more precisions for demands on resources and less overestimation for them.

iv. Decreasing the number of required physical machines results in increasing of net profits and in decreasing of penalties (see Figure 6.19). Decreasing number of required physical machines results in less costs while all tasks can be fitted.

v. PCH/PCH' use less physical machines than the existing configuration of Google traces [95]; the maximal amelioration is 37%, the minimal is about "0%" while the average amelioration is nearly "13,75%" (see Figure 6.20).

vi. PCH/PCH' find physical machines that take into account the quantity of needed resources for live migration of virtual machines hosting live tasks.

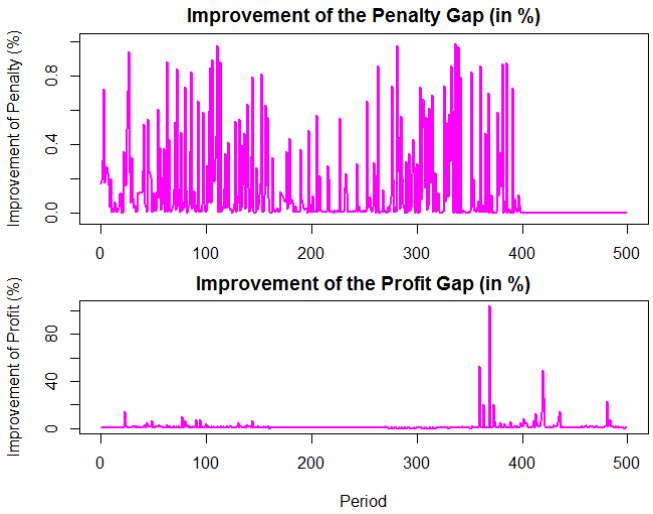

Figure 6.19: Almost all the penalty and profit gaps are in favor to PCH/PCH' and against the existing configuration of Google traces from 2011 [95].

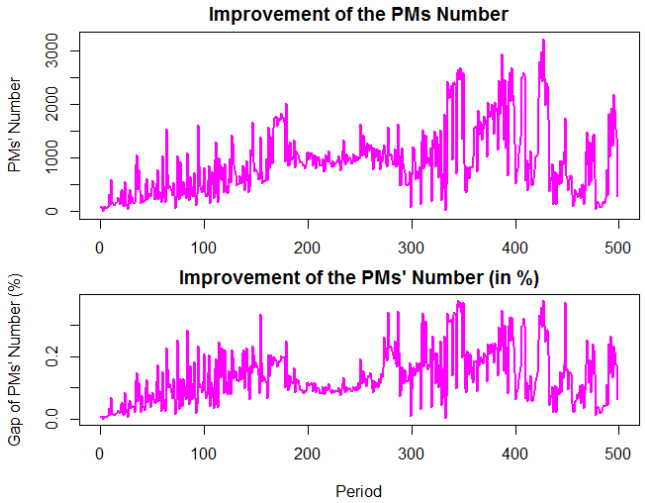

Figure 6.20: Almost all the number of required PMs by PCH/PCH' is less than the number of PMs used by the existing configuration of Google traces from 2011 [95].

6.8 Conclusions and Future Work

Virtualization and server consolidation become important subjects of optimization research because they increase the efficiency and profitability of data centers; therefore, placement and live migration of VMs are used to optimize these NP-Complete problems of load balancing using VMs.

In this paper, we studied the NP-hard problems of live placement of interdependent virtual machines. These problems aim to (i) optimize the penalty centric-problem by minimizing the overall penalty of a virtual cloud data center while respecting its VMs' interdependency constraints; and (ii) optimize the net income of the profit-centric problem by maximizing its overall profit while respecting its SLAs. We tested our approaches with Google traces from 2011 [95]. To solve these problems, we first preprocessed data in the data center to find its trends; then, we formulated the problems mathematically using the models SLV and SLV'. We then proposed algorithms and heuristics PCH and PCH' to approximate these mathematical models for large-sized instances. The extensive simulation results, based on Google traces [95], show that : (i) taking into account negative task correlations optimizes workloads; (ii) there are some task classes in a data center that have higher impacts on trends and on the maximal size of found solutions (i.e. number of required physical machines); (iii) our approximation heuristics (PCH/PCH') find good solutions compared to solutions described in Google traces from 2011 [95]; (iv) our approximation heuristics (PCH/PCH') find solutions in less than $RunningDelay\_max$ for large-sized instances up to K_5 physical machines, $4 * K_5$ virtual machines (see Table 6.3), and many tasks (i.e. more than 20,000 tasks); (v) our approximation heuristics (PCH/PCH') find solutions that takes into account the resources needed by physical machines for live migration of their virtual machines; and (vi) it is important to analyze workload trends periodically to adjust parameters of our algorithms for the VM live placement problems. This study can be a base solution to be extended to other types of VMs' load-balancing optimizations. These future works would find derived SLV and SLV' heuristics considering various heterogeneous technical characteristics of VMs

CHAPITRE 7 DISCUSSION GÉNÉRALE

Dans ce chapitre, nous répondons aux questions de recherche posées dans le chapitre d'introduction. Ensuite, nous discutons de l'approche méthodologique utilisée par rapport à d'autres choix possibles, pour finir par discuter nos résultats et de leur portée.

7.1 Synthèse des travaux

Le but principal de cette thèse est de développer un cadre global de placement d'un ensemble de VMs interdépendantes. Ce placement de VMs peut s'effectuer tout au long de leur utilisation (c.-à-d. lors de leur première installation et postérieurement). Notre étude traite donc ce problème NP-difficile de placement de VMs lors de leur première installation [163]. Ensuite, elle traite du problème de maintien d'une bonne qualité globale de l'ensemble de VMs lors de leur replacement à l'aide de migration en temps réel [164]. Enfin, cette étude traite la consolidation des services virtuels hébergés par des VMs en utilisant, comme base de tests, la configuration existante des services et des machines physiques décrites dans des fichiers des traces de Google [165].

Les travaux présentés dans cette thèse font partie de trois articles, dont deux publiés dans des conférences scientifiques après leurs évaluations par des chercheurs scientifiques. Le troisième article est soumis dans un journal scientifique. Chacun de ces articles répond à un de nos trois objectifs de recherche cités dans le premier chapitre d'introduction. Nous reformulons ces réponses dans les paragraphes qui suivent.

Concernant la première question de recherche, notre étude a démontré la faisabilité d'une planification optimale des machines virtuelles interdépendantes et de leurs ressources partagées. Cela est réalisé par une modélisation mathématique ayant une « fonction objectif » à minimiser et des contraintes à respecter.

Concernant la deuxième question de recherche, notre étude a démontré la possibilité de réaliser des migrations en temps réel de VMs interdépendantes tout en assurant une bonne qualité de service et respectant des contrats de niveaux de service dans un centre virtuel de données. Ceci est réalisé par une modélisation « multi-objectifs » de notre problème de migration, ramené à une modélisation « mono-objectif » et testé par un simulateur de réseau de VMs.

Concernant la troisième question de recherche, notre étude a démontré la possibilité de trouver des solutions de consolidation des services sur des machines virtuelles interdépendantes tout en maximisant le profit net total et en diminuant la pénalité globale due aux non-respects des SLAs et en maintenant une bonne qualité globale des services virtuels.

Ainsi, grâce aux deux premières questions de recherche posées au chapitre d'introduction, nous avons rédigé deux articles de conférence acceptés et présentés dans cette thèse; de plus, nous avons soumis un troisième article de journal répondant à la troisième question de recherche de cette thèse.

Dans la section suivante, nous discutons de la démarche méthodologique utilisée afin d'évaluer et valider nos modèles.

7.2 Méthodologie

Le but principal de cette thèse est de concevoir un modèle global de la planification, de la consolidation, et de la migration des machines virtuelles interdépendantes; pour cela, nous avons utilisé la méthodologie décrite ci-dessous.

Concernant la première question de recherche de cette thèse, nous avons défini un modèle global de planification de plusieurs machines virtuelles. Nous l'avons vérifié et validé en le testant sur un nombre limité de VMs. Ensuite, nous avons utilisé le logiciel mathématique CPLEX comme solveur, utilisant la méthode de la programmation en nombres entiers mixte, pour notre modèle mathématique; ensuite, nous avons développé une heuristique de recherche taboue afin de trouver de bonnes solutions dans des délais raisonnables pour des milliers de machines virtuelles.

Concernant la deuxième question de recherche de cette thèse, nous avons développé un modèle mathématique « multi-objectifs », relaxé par un modèle « mono-objectif », testé et validé par le solveur CPLEX, pour assurer une bonne qualité de service et le respect des contrats de service lors de la migration de plusieurs VMs interdépendantes. Ces migrations doivent supporter la continuité et la fluidité de leurs services. Pour valider la performance de notre modélisation, nous avons simulé un réseau d'un ensemble de VMs hébergées sur des machines physiques.

Concernant la troisième question de recherche de cette thèse, nous avons développé un modèle mathématique et une heuristique d'approximation de recherche taboue pour la consolidation des

services virtuels d'un ensemble des VMs tout en : (i) respectant leurs contraintes d'interdépendance et les limites de capacité des ressources de leurs machines physiques hôtes; (ii) minimisant les pénalités aux SLAs; (iii) maintenant une bonne qualité globale de service; et (iv) minimisant le nombre de machines physiques hôtes nécessaires. Pour valider les performances de notre heuristique, nous les avons comparés avec la configuration existante d'un ensemble des services déployés et bien décrits dans les traces de Google [95]. Comme attendu, nous avons constaté que les solutions de notre heuristique sont de qualité meilleure que nos références de comparaison.

Lors des traitements de nos trois questions de recherche, nous avons utilisé le solveur CPLEX pour des raisons de performance et de disponibilité sur nos ordinateurs. Ce solveur nous permet d'avoir des résultats exacts pour des problèmes NP-difficiles avec de petits ensembles, mais il prend énormément de temps pour résoudre des problèmes NP-difficiles avec des grands ensembles; de même, CPLEX donne des erreurs de dépassement de mémoire avec des ensembles avec des milliers de machines virtuelles.

Nos heuristiques d'approximation se basent sur nos modèles mathématiques afin de trouver des réponses presque aussi exactes que celles des solveurs mathématiques, mais dans de meilleurs délais pour des grands ensembles. Nous pourrions utiliser plusieurs types d'heuristiques autres que les heuristiques de recherche taboue; de même, nous pourrions intégrer CPLEX dans nos heuristiques pour des sous-configurations avec des petits ensembles.

Lors des traitements de nos questions de recherche détaillées dans cette thèse, nous avons utilisé comme base de comparaison des techniques décrites dans la littérature scientifique conçues pour des VMs indépendantes et non pas interdépendantes. Cela est dû au fait que les problèmes que nous avons modélisés, n'ont pas été traités auparavant selon notre revue de littérature. Ainsi, aucun travail n'a traité à date le cas spécifique de l'optimisation des migrations en temps réel simultanées de plusieurs machines virtuelles. Par conséquent, il est difficile pour nous d'utiliser des études scientifiques antérieures comme base de comparaison de nos résultats de planification, de consolidation, et de migration simultanée de plusieurs VMs qui peuvent être interdépendantes ou non. De même, comme nous n'avons pas d'information sur les interdépendances entre les machines virtuelles, nous avons générer aléatoirement des interdépendances entre les services testés.

En somme, nous avons utilisé des choix méthodologiques de modélisation et de simulation qui nous permettent de vérifier et de valider nos solutions. Ainsi, premièrement, nous modélisons les problèmes de planification, de consolidation et de migration en temps réel des VMs, en utilisant des modèles mathématiques munis de plusieurs équations mathématiques. Afin de résoudre ces modèles mathématiques complexes, nous avons recours aux algorithmes de résolution exacte permettant d'avoir des solutions optimales, procurant ainsi des valeurs de référence excellentes. Ensuite, nous utilisons des heuristiques pour leur trouver des approximations dans des délais raisonnables pour des grands jeux de tests.

Ainsi, nous concluons notre discussion concernant notre méthodologie, et nous procédons à l'analyse de nos résultats obtenus.

7.3 Analyse des résultats

Nous sommes satisfaits des résultats obtenus lors de cette étude. L'approche poursuivie pour leur obtention en est l'origine. En effet, nous utilisons une approche globale de migration de ces VMs intégrant leurs diverses contraintes d'interdépendance au lieu de considérer individuellement la migration de chacune de ces VMs dans un contexte de migration simultanée de plusieurs VMs interdépendantes. Nos résultats montrent que notre approche globale fournit de meilleurs résultats que les approches dites classiques basées sur un traitement individuel de chaque VM indépendamment des autres VMs. À notre avis, notre étude ouvre une nouvelle voie dans le domaine de la recherche scientifique appliquée aux problèmes de migration de machines virtuelles, et elle a des incidences positives sur les aspects technologiques et économiques des utilisations des réseaux virtuels.

Concernant la qualité de nos résultats, il est utile de noter leur pertinence par rapport aux résultats d'autres approches définies dans la littérature scientifique traitant des VMs non interdépendantes. Comme indiqué dans les chapitres précédents, nos modèles mathématiques et nos heuristiques génèrent des résultats comparables à ceux de l'approche détaillée dans la littérature scientifiques tout en respectant les contraintes d'interdépendance. De même, nos heuristiques fournissent en moyenne des résultats meilleurs que celui de la configuration détaillée dans les traces Google décrivant les traces d'un très grand nombre de VMs.

Finalement et vu que les algorithmes mathématiques évoluent rapidement et profitent aussi des améliorations des puissances de calcul des ordinateurs, nous sommes optimistes de l'évolution et des améliorations futures de nos algorithmes et de nos heuristiques; ainsi, ces heuristiques pourront s'approcher plus des solutions optimales.

Ainsi, après avoir explicité le bien fondé et la qualité de nos travaux, nous présentons dans la section suivante les conclusions sur notre étude doctorale.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Ce chapitre présente un résumé des travaux réalisés lors de cette thèse. Ainsi, nous présentons une synthèse de nos contributions; ensuite, nous montrons les limites et les contraintes de nos travaux. Enfin, nous présentons certaines recommandations pour de nouvelles pistes de recherche.

8.1 Synthèse de travaux

Selon notre constat, nous estimons que notre travail est un élément à valeur ajoutée pour les recherches scientifiques appliquées aux réseaux virtuels. Notre travail est original par rapport à la majorité des travaux réalisés jusqu'à présent dans le domaine de la planification, la consolidation, et la migration en temps réel des machines virtuelles, car il traite de la planification, la consolidation, et des migrations groupées d'un ensemble de VMs et pas seulement d'une seule VM à la fois; jusqu'à présent et à notre connaissance, il n'y a pas beaucoup de travaux qui optimisent la planification, la consolidation, et la migration globale simultanée des VMs parallèles interdépendantes. La majorité des travaux actuels se concentrent sur la planification, la consolidation, et la migration non simultanée des VMs non interdépendantes.

Pour cette raison, nous proposons un modèle général capable de traiter simultanément la planification, la consolidation, et la migration de n VMs (avec $n \geq 2$) de divers types. Étant donné la difficulté de ce problème, les techniques actuelles décomposent le problème de planification et de migration simultanée de plusieurs VMs en plusieurs sous-problèmes de planification et de migration individuelles de chacune des VMs; de même, ces techniques actuelles manquent souvent de traiter la planification et la migration des ressources partagées (entre des VMs interdépendantes). Cette décomposition a le fâcheux désavantage de dégrader la qualité des solutions obtenues. Notre travail est original puisqu'il apporte des solutions pour ces types de dégradation; ainsi, nos modèles mathématiques et heuristiques proposent des résolutions globales pour les planifications, consolidations, et migrations simultanées des machines virtuelles, en plus d'être capable de s'adapter à divers types de VMs comme des machines virtuelles interdépendantes. Comme nous l'avons présenté auparavant, nos modèles peuvent s'adapter facilement aux planifications, consolidations et migrations des VMs indépendantes, des VMs interdépendantes, ainsi qu'à plusieurs autres types de VMs.

Notre travail donne lieu à plusieurs contributions dans le domaine de réseaux virtuels, et plus particulièrement en ce qui concerne la planification, la consolidation et la migration en temps réel des machines virtuelles coopératives et parallèles. Notre but principal étant la modélisation d'un cadre global de planification, de consolidation et de migration à temps réel des VMs.

Voici, un résumé des contributions de notre étude :

- modélisation mathématique pour traiter la qualité de service lors de la migration en temps réel des machines virtuelles interdépendantes. De même, cette modélisation prend en considération les ressources nécessaires d'un réseau afin de respecter des contrats de niveaux de service. Cette modélisation mathématique est ensuite utilisée dans plusieurs simulations pour montrer la faisabilité de respecter des contrats SLAs et maintenir la QoS des applications interdépendantes lors de la migration en temps réel de leurs VMs hôtes [163];
- modélisation d'une approche globale pour traiter les migrations en temps réel des machines virtuelles pour permettre une amélioration « multi-objectifs », relaxée à une amélioration « mono-objectifs », de migration en temps réel d'un ensemble de VMs interdépendantes. Cette modélisation, via des équations mathématiques, des problèmes de migration en temps réel simultanées de plusieurs VMs, nous permet d'améliorer la qualité globale, réduire les temps de migration de VMs, minimiser leur temps d'arrêt de service, de maximiser le profit net total, et diminuer les pénalités aux contrats de service tout en respectant les contraintes d'interdépendance de ces VMs [164]. Cette modélisation est effectuée à l'aide de l'outil CPLEX qui fournit des solutions optimales. Les résultats de cette étude montrent que cette approche « multi-objectifs », relaxée par une approche « mono-objectif », permet de réduire les pénalités aux SLAs lors des migrations en temps réel de VMs;
- modélisation par des équations mathématiques d'une approche globale pour améliorer la qualité globale des services d'un ensemble de VMs via la diminution de sa pénalité globale aux contrats de niveaux de service [165], et à la diminution du nombre de machines physiques hôtes nécessaires pour héberger ses services. Cette modélisation respecte les contraintes d'interdépendance des VMs, et elle est testée, validée et comparée à la configuration utilisée dans les traces de Google afin de démontrer ses performances.

L’originalité de notre recherche réside dans la prise en considération des VMs interdépendantes et parallèles pour l’étude de la planification, consolidation, et migration en temps réel des VMs. Jusqu’à présent et selon notre revue de littérature, peu d’études se sont attardées à l’étude des VMs parallèles et interdépendantes migrées en temps réel et utilisant des ressources partagées. Par conséquent et selon notre revue de littérature, notre étude est innovatrice, car elle cherche à optimiser la planification, la consolidation et la migration simultanée de plusieurs VMs afin de répondre à un ou plusieurs objectifs et respecter leurs différentes contraintes.

Un deuxième point en faveur de l’innovation de notre étude est l’utilisation des méthodes heuristiques (à l'aide de la méta-heuristique de recherche taboue) approximant, dans des délais raisonnables, nos modèles mathématiques de résolution de la planification, consolidation et migration de VMs. Ce deuxième point de contribution est à la fois original, utile et utilisable par les responsables de planification, consolidation, et migration des VMs.

Un troisième et dernier point intéressant concernant l’originalité de notre travail consiste à l’utilisation de la totalité, et non seulement d'un échantillon, des données en notre possession pour prendre les décisions sur les placements des services virtuels, de leurs VMs et machines physiques hôtes; cette approche permet aussi de trouver de bonnes solutions dans des délais très acceptables; ces solutions maintiennent une bonne qualité de service du réseau de VMs, minimisent les pénalités sur les SLAs, maximisant le profit net total, et réduisent le nombre des machines physiques nécessaires pour fournir des services virtuels.

Dans ce qui suit, nous présentons les limitations de nos travaux, pour finir par exposer nos recommandations pour des travaux futurs.

8.2 Limitations des solutions proposées

Nous détaillons, dans cette section, les limites et les contraintes de notre étude. Comme tout travail de recherche, notre étude connait aussi des limitations à savoir:

- des difficultés à résoudre, dans des délais raisonnables, les modèles mathématiques modélisant les problèmes de planification, consolidation et migrations simultanées de plusieurs VMs interdépendantes;

- des difficultés de prédire le trafic dans des réseaux virtuels composés de plusieurs VMs sachant que ces VMs envoient simultanément plusieurs et divers messages constituant un trafic dynamique changeant continuellement en fonction du temps;

- des difficultés de faire le tour de tous les types existants de VMs sur le marché des réseaux virtuels et de les tester avec des modèles mathématiques et des heuristiques;

- des difficultés de générer de bonnes bornes inférieures pour la planification et la consolidation d'un nombre très élevé de VMs à cause du dépassement de la capacité de mémoire de l'ordinateur;

- des difficultés de trouver les meilleurs techniques de résolution de l'approche « multi-objectifs » permettant de trouver des solutions pour des migrations en temps réel d'un très grand nombre de VMs. Plusieurs techniques pourraient le réaliser, mais elles devraient être flexibles et modulables; ainsi, il est préférable de tester plusieurs techniques et de les comparer ensemble afin d'en trouver les meilleures.

8.3 Améliorations futures

Après la description, détaillée dans la section précédente des limites de notre étude, nous proposons les pistes suivantes d'amélioration future:

- afin de rendre une solution, de planification et de consolidation de VMs, utile pour l'expansion des réseaux existants, il serait intéressant d'étudier la possibilité d'intégrer dans ses modèles mathématiques les contraintes des emplacements des VMs déjà existantes;

- afin de rendre l'approche des solutions d'optimisation de VMs « multi-objectifs » avec un très grand nombre de VMs, il serait utile d'utiliser des agents locaux distribués pour rendre les calculs modulables et flexibles;

- afin d'améliorer les résultats et les temps de calcul des solutions d'optimisation de VMs, il serait utile de tester des techniques de la programmation par contraintes ou des heuristiques autres que les recherches locales et les recherches taboues; par exemple, il serait intéressant

de les tester avec des heuristiques hybrides intégrant à la fois des calculs exacts et des heuristiques comme la recherche locale, la recherche taboue, la colonie de fourmis, etc.

- afin d'utiliser des solutions d'optimisation de VMs avec des périphériques mobiles comme des tablettes et des téléphones intelligents, il serait utile de leur intégrer les notions des contraintes de la mobilité de leurs machines physiques hôtes;

- afin d'utiliser des solutions d'optimisation des VMs d'une manière sécurisée, il serait utile de leur intégrer des solutions déjà existantes de sécurité informatique comme le chiffrement et déchiffrement du contenu des VMs ou bien de développer leurs propres techniques sécuritaires;

- afin de prendre en considération la survivabilité des réseaux de VMs, il serait utile d'intégrer le traitement préventif des erreurs probables afin de rendre un réseau de VMs plus robuste face aux pannes;

- afin de mieux prendre en considération les capacités en ressources physiques d'un réseau virtuel, il serait utile de considérer aussi les contraintes des hôtes physiques hébergeant des machines virtuelles interdépendantes en plus des contraintes des VMs elles-mêmes;

- afin de mieux prendre en considération l'évolutivité continue des utilisations des ressources d'un réseau virtuel, il serait utile de le simuler avec des systèmes des automates cellulaires ou bien avec des systèmes multi-agents en utilisant des logiciels comme « Breve » [166], « Starlogo TNG », ou « Netlogo » [167] dérivé du « Starlogo » développé par Dr. Mitchel Resnick.

RÉFÉRENCES

[1] T. Wood, K. Ramakrishnan, J. Van Der Merwe, and P. Shenoy, "Cloudnet: A platform for optimized wan migration of virtual machines," *University of Massachusetts Technical Report TR-2010-002*, 2010.

[2] Z. Xu, S. Di, W. Zhang, L. Cheng, and C.-L. Wang, "Wavnet: Wide-area network virtualization technique for virtual private cloud," in *International Conference on Parallel Processing*, 2011, pp. 285-294.

[3] F. Romero and T. J. Hacker, "Live migration of parallel applications with openvz," in *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, 2011, pp. 526-531.

[4] K. Ye, X. Jiang, D. Huang, J. Chen, and B. Wang, "Live migration of multiple virtual machines with resource reservation in cloud computing environments," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 267-274.

[5] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 50-55, 2008.

[6] A. Chazalet, "Service level checking in the cloud computing context," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 297-304.

[7] V. Sarathy, P. Narayan, and R. Mikkilineni, "Next generation cloud computing architecture: Enabling real-time dynamism for shared distributed physical infrastructure," in *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, 2010, pp. 48-53.

[8] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *Services Computing, IEEE Transactions on*, vol. 3, pp. 266-278, 2010.

[9] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*: John Wiley & Sons, Inc., 1990.

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, pp. 273-286.

[11] M. St-Hilaire, S. Chamberland, and S. Pierre, "Global and sequential approaches for UMTS network design," in *Communications and Computer Networks: Third IASTED International Conference Proceedings*, 2005.

[12] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, pp. 599-616, 2009.

[13] A. Barnoschi, "Backup and Disaster Recovery For Modern Enterprise," in *5th International Scientific Conference, Business and Management*, Lithuvania, 2008.

[14] G. Chen, Q. Kong, J. Etheridge, and P. Foster, "Integrated TMN service management," *Journal of Network and Systems Management*, vol. 7, pp. 469-493, 1999.

[15] M. Dilman and D. Raz, "Efficient reactive monitoring," *Selected Areas in Communications, IEEE Journal on*, vol. 20, pp. 668-676, 2002.

[16] A. Beugnard, J.-M. Jézéquel, N. Plouzeau, and D. Watkins, "Making components contract aware," *Computer*, vol. 32, pp. 38-45, 1999.

[17] L. Bodenstaff, A. Wombacher, M. Reichert, and M. C. Jaeger, "Monitoring dependencies for slas: The mode4sla approach," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, 2008, pp. 21-29.

[18] A. Keller and H. Ludwig, "The WSLA framework: Specifying and monitoring service level agreements for web services," *Journal of Network and Systems Management*, vol. 11, pp. 57-81, 2003.

[19] Layer 7 Technologies, "SecureSpan XML Networking Gateway Advanced Service Management and Mediation," 2008.

[20] Q. Wang, Y. Liu, M. Li, and H. Mei, "An online monitoring approach for web services," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, 2007, pp. 335-342.

[21] A. Sahai, S. Graupner, V. Machiraju, and A. Van Moorsel, "Specifying and monitoring guarantees in commercial grids through SLA," in *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, 2003, pp. 292-299.

[22] D. Ameller and X. Franch, "Service level agreement monitor (SALMon)," in *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, 2008, pp. 224-227.

[23] G. Wiederhold, "Mediators in the architecture of future information systems," *Computer*, vol. 25, pp. 38-49, 1992.

[24] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, *et al.*, "Web services agreement specification (WS-Agreement)," in *Open Grid Forum*, 2007, p. 216.

[25] Y. Julien and J. Sobrino, "CloudSim: A fair benchmark for comparison of methods for times series reconstruction from cloud and atmospheric contamination," in *Analysis of Multitemporal Remote Sensing Images (Multi-Temp), 2015 8th International Workshop on the*, 2015, pp. 1-4.

[26] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Applied Soft Computing*, vol. 11, pp. 4135-4151, 2011.

[27] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958-2008*, ed: Springer, 2010, pp. 105-132.

[28] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations research*, vol. 11, pp. 972-989, 1963.

[29] P. Van Hentenryck, *Constraint satisfaction in logic programming*, vol. 5: MIT press Cambridge, 1989.

[30] K. Apt, *Principles of constraint programming*: Cambridge University Press, 2003.

[31] D. Sam-Haroud and B. Faltings, "Consistency techniques for continuous constraints," *Constraints,* vol. 1, pp. 85-118, 1996.

[32] P. Van Hentenryck, V. Saraswat, and Y. Deville, "Design, implementation, and evaluation of the constraint language cc (FD)," *The Journal of Logic Programming,* vol. 37, pp. 139-164, 1998.

[33] IBM Inc., "IBM ILOG CPLEX Optimizer," 2012.

[34] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research,* vol. 13, pp. 533-549, 1986.

[35] O. Brun and J.-M. Garcia, "Ressource allocation in communication networks," in *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, 2002, pp. 229-233.

[36] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, 1991, pp. 134-142.

[37] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politecnico di Milano, Italy,* 1992.

[38] L. F. Ibrahim, O. Metwaly, A. Kapel, and A. Ahmed, "Enhancing the behavior of the ant algorithms to solving network planning problem," in *Software Engineering Research, Management and Applications, 2005. Third ACIS International Conference on*, 2005, pp. 250-255.

[39] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of statistical physics,* vol. 34, pp. 975-986, 1984.

[40] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research,* vol. 24, pp. 1097-1100, 1997.

[41] J. Dréo, A. Pétrowski, É. D. Taillard, and P. Siarry, *Métaheuristiques pour l'optimisation difficile,* 2003.

[42] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*, ed: Springer, 2011, pp. 760-766.

[43] W. Bateson and G. Mendel, *Mendel's principles of heredity*: Courier Corporation, 2013.

[44] C. Darwin and G. G. Simpson, *The origin of species by means of natural selection: Or, The preservation of favoured races in the struggle for life*: Collier Books New York, 1962.

[45] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press, 1975.

[46] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operations research,* vol. 63, pp. 511-623, 1996.

[47] E. A. Silver, R. Victor, V. Vidal, and D. de Werra, "A tutorial on heuristic methods," *European Journal of Operational Research,* vol. 5, pp. 153-162, 1980.

[48] J. Puchinger and G. R. Raidl, *Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification*: Springer, 2005.

[49] G. R. Raidl and J. Puchinger, "Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization," in *Hybrid metaheuristics*, ed: Springer, 2008, pp. 31-62.

[50] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, *et al.*, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 91-98.

[51] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, pp. 392-403, 2001.

[52] B. Mareschal, "Aide à la décision multicritère: Développements récents des méthodes PROMETHEE," *Cahiers du Centre d'études de recherche opérationnelle*, vol. 29, pp. 175-214, 1987.

[53] Y. O. Yazır, C. Matthews, R. Farahbod, A. Guitouni, S. Neville, S. Ganti, *et al.*, "Dynamic and Autonomous Resource Management in Computing Clouds through Distributed Multi Criteria Decision Making," *University of Victoria, Department of Computer Science, Tech. Rep. DCS-334-IR*.

[54] J. G. Hansen and A. K. Henriksen, "Nomadic operating systems," Master's thesis, Dept. of Computer Science, University of Copenhagen, Denmark, 2002.

[55] M. R. Hines, U. Deshpande, and K. Gopalan, "Post-copy live migration of virtual machines," *ACM SIGOPS operating systems review*, vol. 43, pp. 14-26, 2009.

[56] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "IP over P2P: enabling self-configuring virtual IP networks for grid computing," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, pp. 49-59.

[57] S. J. Hussain and G. Ahmed, "A comparative study and analysis of PVM and MPI for parallel and distributed systems," in *Information and Communication Technologies, 2005. ICICT 2005. First International Conference on*, 2005, pp. 183-187.

[58] D. S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process migration," *ACM Computing Surveys (CSUR)*, vol. 32, pp. 241-299, 2000.

[59] S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The design and implementation of Zap: A system for migrating computing environments," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 361-376, 2002.

[60] Cisco Inc., "Cisco Active Network Abstraction," 2014.

[61] J. Networks, "Configuration and Diagnostic Automation Guide," 2014.

[62] R. Wojtczuk, "Adventures with a certain Xen vulnerability (in the PVFB backend)," *Message sent to bugtraq mailing list on October 15th*, 2008.

[63] A. Mirkin, A. Kuznetsov, and K. Kolyshkin, "Containers checkpointing and live migration," in *Proceedings of the Linux Symposium*, 2008, pp. 85-90.

[64] M. Richmond and M. Hitchens, "A new process migration algorithm," *SIGOPS Oper. Syst. Rev.*, vol. 31, pp. 31-42, 1997.

[65] K. Chard, S. Caton, O. Rana, and K. Bubendorfer, "Social cloud: Cloud computing in social networks," in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 99-106.

[66] French ANR RNTL, "SemEUsE cooperative research project."

[67] European Celtic, "SERVERY cooperative research project," SERVERY project consortium, 2011.

[68] OW2 Consortium and Bull, "JOnAS: Open Source Java EE Application Server," 2010.

[69] OW2 consortium, "Dream: a component-based communication framework," 2009.

[70] G. Blair, T. Coupaye, and J.-B. Stefani, "Component-based architecture: the Fractal initiative," *Annals of Telecommunications*, vol. 64, pp. 1-4, 2009.

[71] Sun Microsystems Corporation, "Java Management Extensions (JMX) Technology," 2010.

[72] Apache Software Foundation, "Wicket (version 1.4.5)," 2010.

[73] OW2 Consortium and Bull, "JASMINe: The Smart Tool for your SOA Platform Management," 2010.

[74] A. Diaconescu and B. Dillenseger, "A componentbased architectural approach to autonomic management of complex systems-Generic Monitoring for Autonomic Management Systems," *IAWASM, INRIA Rhone-Alpes, France,* 2006.

[75] B. Debbabi, "Framework orienté-service de médiation de données," *Master report, Université de Grenoble*, vol. 24, 2009.

[76] A. Velte and T. Velte, *Microsoft virtualization with Hyper-V*: McGraw-Hill, Inc., 2009.

[77] D. Chisnall, *The definitive guide to the xen hypervisor*: Pearson Education, 2008.

[78] Amazon, "Building facebook applications on aws website," 2012.

[79] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, M. Arlitt, *et al.*, "Facebook meets the virtualized enterprise," in *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE,* 2008, pp. 286-292.

[80] L. Peterson and T. Roscoe, "The design principles of PlanetLab," *ACM SIGOPS operating systems review*, vol. 40, pp. 11-16, 2006.

[81] K. Chard, W. Tan, J. Boverhof, R. Madduri, and I. Foster, "Wrap scientific applications as WSRF grid services using gRAVI," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 83-90.

[82] D. Neumann, J. Stoesser, A. Anandasivam, and N. Borissov, "SORMA–building an open grid market for grid resource allocation," in *Grid Economics and Business Models*, ed: Springer, 2007, pp. 194-200.

[83] N. Borissov, S. Caton, O. Rana, and A. Levine, "Message protocols for provisioning and usage of computing services," in *Grid Economics and Business Models*, ed: Springer, 2009, pp. 160-170.

[84] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, *et al.*, "Job submission description language (jsdl) specification, version 1.0," in *Open Grid Forum, GFD*, 2005.

[85] K. Chard and K. Bubendorfer, "Using secure auctions to build a distributed meta-scheduler for the grid," *Market Oriented Grid and Utility Computing*, pp. 569-588, 2009.

[86] A. Petitet, R. Whaley, J. Dongarra, and A. Cleary, "HPL–a portable implementation of the high–performance Linpack benchmark for distributed–memory computers," 2008.

[87] M. Luisier and G. Klimeck, "Performance limitations of graphene nanoribbon tunneling FETS due to line edge roughness," in *IEEE Device Res. Conf*, 2009, pp. 201-202.

[88] M. Luisier and G. Klimeck, "A multi-level parallel simulation approach to electron transport in nano-scale transistors," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, 2008, pp. 1-10.

[89] Gartner, "Gartner's 2008 Data Center Conference Instant Polling Results: Virtualization Summary," 2009.

[90] S. Sud, R. Want, T. Pering, K. Lyons, B. Rosario, and M. X. Gong, "Dynamic migration of computation through virtualization of the mobile platform," *Mobile Networks and Applications*, vol. 17, pp. 206-215, 2012.

[91] F. Checconi, T. Cucinotta, and M. Stein, "Real-time issues in live migration of virtual machines," in *Euro-Par 2009–Parallel Processing Workshops*, 2009, pp. 454-466.

[92] D. Breitgand, G. Kutiel, and D. Raz, "Cost-Aware Live Migration of Services in the Cloud," in *SYSTOR*, 2010.

[93] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based optimization of power and migration cost in cloud computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, 2012, pp. 172-179.

[94] Intel Inc. (2016). *Desktop Processors*. Available: http://ark.intel.com/

[95] Google Inc. (2011). *Google Traces of 2011*. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_2

[96] S.-E. Benbrahim, A. Quintero, and M. Bellaiche. (2014). *On the Design of Virtual Networks Optimized for Interdependent Virtual Machines*. Available: https://drive.google.com/open?id=0BxwQIPlgCMzCNlk3OU9DTGFmN3M

[97] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Resource leasing and the art of suspending virtual machines," in *High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on*, 2009, pp. 59-68.

[98] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, *et al.*, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 3-12, 2003.

[99] U. Bellur and C. S. Rao, "Optimal placement algorithms for virtual machines," *arXiv preprint arXiv:1011.5064*, 2010.

[100] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, 2009, pp. 17-25.

[101] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 44-51.

[102] V. Goswami, S. S. Patra, and G. Mund, "Performance analysis of cloud with queue-dependent virtual machines," in *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on*, 2012, pp. 357-362.

[103] B. Urgaonkar, A. L. Rosenberg, and P. Shenoy, "Application placement on a cluster of servers," *International Journal of Foundations of Computer Science,* vol. 18, pp. 1023-1041, 2007.

[104] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, pp. 2923-2938, 2009.

[105] S. Mehta and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *Network Operations and Management Symposium*, 2008, pp. 363-370.

[106] L. Shi, B. Butler, R. Wang, D. Botvich, and B. Jennings, "Optimal placement of virtual machines with different placement constraints in IAAS clouds," in *ICT and Energy Efficiency and Workshop on Information Theory and Security (CIICT 2012), Symposium on*, 2012, pp. 35-40.

[107] M. St-Hilaire, S. Chamberland, and S. Pierre, "Uplink UMTS network design—an integrated approach," *Computer Networks*, vol. 50, pp. 2747-2761, 2006.

[108] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, pp. 1397-1420, 2012.

[109] F. Glover and E. Taillard, "A user's guide to tabu search," *Annals of operations research*, vol. 41, pp. 1-28, 1993.

[110] F. Glover, "Tabu search—part II," *ORSA Journal on computing*, vol. 2, pp. 4-32, 1990.

[111] ILOG Inc., "10.0 User's manual," 2006.

[112] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *Quality of Service—IWQoS 2003*, ed: Springer, 2003, pp. 381-398.

[113] G. Wang and N. T. Eugene, "The impact of virtualization on network performance of amazon ec2 data center," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1-9.

[114] L. Columbus, "Roundup of cloud computing forecasts and market estimates," *A Passion*, 2015.

[115] Cisco Inc., "Cisco's Cloud Strategy for Cloud Providers," 2014.

[116] Datadog Inc., "The Top 5 AWS EC2 Performance Problems," 2013.

[117] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, *et al.*, "Data center network virtualization: A survey," *Communications Surveys & Tutorials, IEEE,* vol. 15, pp. 909-928, 2013.

[118] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 17-29, 2008.

[119] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review,* vol. 41, pp. 38-47, 2011.

[120] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1-9.

[121] N. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*, 2009, pp. 783-791.

[122] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei, "Are all games equally cloud-gaming-friendly? an electromyographic approach," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1-6.

[123] S. Shi, K. Nahrstedt, and R. Campbell, "Distortion over latency: Novel metric for measuring interactive performance in remote rendering systems," in *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, 2011, pp. 1-6.

[124] P. Chen and M. El Zarki, "Perceptual view inconsistency: an objective evaluation framework for online game quality of experience (QoE)," in *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games*, 2011, p. 2.

[125] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM),* vol. 10, p. 10, 2014.

[126] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing virtual machines to optimize cloud gaming experience," *Cloud Computing, IEEE Transactions on*, vol. 3, pp. 42-53, 2015.

[127] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds," *Cloud Computing, IEEE Transactions on*, vol. 1, pp. 129-141, 2013.

[128] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, 2011, pp. 1-6.

[129] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, pp. 1027-1034, 2011.

[130] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 594-601.

[131] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 237-250.

[132] Y. Li, W. Li, and C. Jiang, "A survey of virtual machine system: Current technology and future trends," in *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, 2010, pp. 332-336.

[133] S. Chiueh and S. Brook, "A survey on virtualization technologies," *RPE Report*, pp. 1-42, 2005.

[134] R. Rose, "Survey of system virtualization techniques," 2004.

[135] M. Missbach and J. Stelzel, "Adaptive Hardware-Infrastrukturen für SAP," ed: SAP Press, 2005.

[136] Q. Zhang, L. Cherkasova, G. Mathews, W. Greene, and E. Smirni, "R-capriccio: A capacity planning and anomaly detection tool for enterprise services with live workloads," in *Middleware 2007*, ed: Springer, 2007, pp. 244-265.

[137] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao, "Resource constrained scheduling as generalized bin packing," *Journal of Combinatorial Theory, Series A*, vol. 21, pp. 257-298, 1976.

[138] C. Chekuri and S. Khanna, "On Multi-Dimensional Packing Problems," in *SODA*, 1999, pp. 185-194.

[139] N. Bansal, A. Caprara, and M. Sviridenko, "Improved approximation algorithms for multidimensional bin packing problems," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, 2006, pp. 697-708.

[140] M. Yue, "A simple proof of the inequality FFD (L)≤ 11/9 OPT (L)+ 1,∀ L for the FFD bin-packing algorithm," *Acta mathematicae applicatae sinica*, vol. 7, pp. 321-331, 1991.

[141] G. Dósa, "The tight bound of first fit decreasing bin-packing algorithm is FFD (I)≤ 11/9OPT (I)+ 6/9," in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, ed: Springer, 2007, pp. 1-11.

[142] A. Kemper and A. Eickler, *Datenbanksysteme: Eine Einführung*: Oldenbourg Verlag, 2011.

[143] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 1269-1272.

[144] M. H. Kutner, C. Nachtsheim, and J. Neter, *Applied linear regression models*: McGraw-Hill/Irwin, 2004.

[145] Y. Chen, S. Alspaugh, and R. H. Katz, "Design insights for MapReduce from diverse production workloads," California University Berkeley Department of Electrical Engineering and Computer Science, 2012.

[146] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An analysis of traces from a production mapreduce cluster," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 94-103.

[147] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from Google compute clusters," *ACM SIGMETRICS Performance Evaluation Review,* vol. 37, pp. 34-41, 2010.

[148] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, 2012, p. 7.

[149] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, "Modeling and synthesizing task placement constraints in Google compute clusters," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, p. 3.

[150] J. Wilkes, "More Google cluster data," *Google research blog, Nov,* 2011.

[151] Q. Zhang, J. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in Google's compute clusters," in *Large Scale Distributed Systems and Middleware Workshop (LADIS'11)*, 2011.

[152] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 5-18.

[153] F. Margot, "Symmetric ILP: Coloring and small integers," *Discrete Optimization,* vol. 4, pp. 40-62, 2007.

[154] P. Brierley, D. Vogel, and R. Axelrod, "Heritage Provider Network Health Prize Round 1 Milestone Prize: How we did it–Team 'Market Makers'," ed, 2011.

[155] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189-1232, 2001.

[156] R. Packages, "Neural Networks Back Propagation Weight Update Rule Neural Network Source Code," 2011.

[157] R. Packages, "Bagged Trees Random Forests Papers randomForest package in R," 2004.

[158] R. Packages, "Linear Models lm function in the R Stats Package Least Squares," 2014.

[159] M. Claypool, "Motion and scene complexity for streaming video games," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, pp. 34-41.

[160] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM,* vol. 49, pp. 40-45, 2006.

[161] M. St-Hilaire, S. Chamberland, and S. Pierre, "A tabu search algorithm for the global planning problem of third generation mobile networks," *Computers & Electrical Engineering,* vol. 34, pp. 470-487, 2008.

[162] F. Glover, "Tabu search-part I," *ORSA Journal on computing,* vol. 1, pp. 190-206, 1989.

[163] S.-E. Benbrahim, A. Quintero, and M. Bellaiche, "On the Design of Large-Sized Cloud Networks Optimized for Live Migration Using Tabu Search Algorithms," in *Modelling Symposium (AMS), 2014 8th Asia*, 2014, pp. 202-206.

[164] S.-E. Benbrahim, A. Quintero, and M. Bellaiche, "Real-Time QoS Issues in Live Migration of Interdependent Virtual Machines," in *Modelling Symposium (AMS), 2014 8th Asia*, 2014, pp. 197-201.

[165] S.-E. Benbrahim, A. Quintero, and M. Bellaiche, "Live Placement of Interdependent Virtual Machines to Optimize Cloud Service Profits and Penalties on SLAs," *IEEE Transactions on Services Computing (Submitted),* 2015.

[166] J. Klein, "Breve: a 3D simulation environment for multi-agent simulations and artificial life," *Hampshire College,* 2005.

[167] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International conference on complex systems*, 2004.

[168] S. R. Pakize and S. M. Khademi, "Comparison Of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing," 2014.

ANNEXE A - ANALYSE MATHÉMATIQUE DE LA PLANIFICATION DES MACHINES VIRTUELLES INTERDÉPENDANTES

Cette annexe détaille le modèle mathématique utilisé pour résoudre le problème NP-difficile de la planification des machines virtuelles interdépendantes; un ensemble de machines virtuelles (VMs) interdépendantes comportent des VMs qui doivent échanger certaines de leurs informations dans des délais prédéfinis; la transmission et la propagation de ces informations doivent respecter certaines contraintes de délais (voir ci-dessous).

Nous modélisons le problème NP-difficile de la planification de ces machines virtuelles interdépendantes (GPV) afin lui trouver une solution quasi-optimale. Notre modélisation utilise la notation définie au Tableau A.1. La fonction objectif f (A. 1) d'un réseau de VMs interdépendantes peut être exprimée de la manière suivante:

$$\min (C_{links} + C_{nodes}) \tag{A.1}$$

Le premier terme C_{links} (A.7) représente le coût des liens entre les nœuds d'un réseau de VMs (VMMs, SDSRVs, MRs, HRs); ces nœuds peuvent être connectés par différents types de liens; le deuxième terme C_{nodes} (A.8) prend en compte les coûts des types des équipements des nœuds placés. La fonction objectif f (A.1) est sujet aux contraintes des délais (A.2) - (A.6) et des contraintes des assignations et des flux (A.9) - (A.29) décrites ci-après. Les contraintes (A.2) - (A.3) montrent que les délais d'échange des informations entre deux VMs interdépendantes doivent être inférieurs à un seuil prédéfini, et elles peuvent être exprimées comme suit:

$$MaxInterDelay_{c_{m1}} \geq InterDependent_{c_{m1},h_1,c_2,h_2} * (Delay_{c_{m1},h_1,i_1,j_1,k_1} + Delay_{c_{m2},h_2,i_2,j_2,k_2} + MR_{InterDelay,k_1k_2}) \quad \forall h_1, \forall i_1, \forall j_1, \forall k_1, \forall c_{m1} \forall h_2, \forall i_2, \forall j_2, \forall k_2, \forall c_{m2} \tag{A.2}$$

$$MaxInterDelay_{c_{m2}} \geq InterDependent_{c_{m1},h_1,c_2,h_2} * (Delay_{c_{m1},h_1,i_1,j_1,k_1} + Delay_{c_{m2},h_2,i_2,j_2,k_2} + MR_{InterDelay,k_1k_2}) \quad \forall h_1, \forall i_1, \forall j_1, \forall k_1, \forall c_{m1} \forall h_2, \forall i_2, \forall j_2, \forall k_2, \forall c_{m2} \tag{A.3}$$

Tableau A.1: Notation et définitions

Notation	Définition
c, c_m, c_h	Respectivement, l'indice du type de service, d'un service partageant des données de sa mémoire, et d'un service ayant un disque virtuel hébergé sur un SDSRV.
C_{memory}	Ensemble des types des applications partageant des données de leur mémoire.
C_{hDisk}	Ensemble des types des applications ayant un disque virtuel hébergé par un SDSRV.
C_{nodes} , C_{links}	Respectivement, le coût total des nœuds placés et le coût total des liens entre ces nœuds.
$Delay_{c,h,i,j,k}$	Délai agrégé de la transmission et propagation des données d'une application de classe c entre VM_het MR_k.
$d_{a,b}$	Distance (en mètres) entre deux nœuds a et b.
$f_{y,c}$, $f_{x,y,c}$, $M_{k,t}$	Respectivement, le flux de type c dirigé vers un nœud y, flux de type c entre x et y, capacité en flux d'un nœud k de type t.
h, i, j,k,l	Respectivement, l'indice d'un VM, VMM, SDSRV, MR, et HR.
$InterDependent_{c_1,h_1,c_2,h_2}$	Variable binaire égale à "1" si l'application du type c_1 placée sur VMM_{h_1}est interdépendante avec une application du type c_2 placée sur VMM_{h_2}; sinon, elle est égale à "0".
K_{c_m}	Taux de transmission d'une application du type c_m.
$MR_{InterDelay,x,y}$	Délai de propagation sur un lien connectant MR_x à MR_y.
$MaxInterDelay_{c_m}$, $TransInterMemory_{c_m}$	Respectivement, le seuil maximal acceptable du délai de propagation, et la quantité de mémoire transférée d'une application du type c_m.
N_{VM}, N_{VMM}, N_{SDSRV}, N_{MR}, N_{HR}	Respectivement, ensemble de VMs, VMMs, SDSRVs, MRs, et HRs.
$PropDelay_{a,b}$	Délai de propagation entre deux nœuds a et b.
s_m	Vitesse de propagation d'un lien du type m liant deux nœuds.
T_{VMM},T_{SDSRV},T_{MR}, T_{HR}	Respectivement, ensemble des types de VMMs, SDSRVs, MRs, et HRs.
$T_{VM,VMM}$,$T_{VMM,SDSRV}$, $T_{SDSRV,MR}$, $T_{SDSRV,HR}$	Respectivement, ensemble des types des liens "VMs vers VMMs", "VMMs vers SDSRVs", "SDSRVs vers MRs", et "SDSRVs vers HRs".
$TransDelay_c$	Délai de transmission d'une application du type c.
$v_{a,b.m}$	Variable binaire égale à "1" si le nœud a est lié au nœud b via un lien du type m; sinon, elle est égale à "0".
$x_{a,t}$	Variable binaire égale à "1" si le nœud a du type t est placé; sinon, elle est égale à "0".
μ_{h,c_m}	Nombre des applications du type c_m hebergées par VM_h.

Notation	Définition
$\alpha_{a,b,m}$	Coût (en dollars/kilomètre) de l'installation d'un lien du type m connectant deux nœuds a et b.
$\beta_{a,t}$	Coût de l'installation d'un nœud a du type t.

Le délai de transfert d'une application, de classe c placée sur VM_h et propagée vers MR_k, est une agrégation du délai de la transmission sur VM_h et de ses délais de propagation sur les liens du réseau. Ces délais de propagation incluent le délai du VM_h vers VMM_i, du VMM_i vers $SDSRV_j$, et du $SDSRV_j$ vers MR_k; cela peut s'exprimer de la manière suivante:

$$Delay_{c,h,i,j,k} = TransDelay_c + PropDelay_{h,i} + PropDelay_{i,j} + PropDelay_{j,k} \qquad \forall h, \forall i, \forall j, \forall k, \forall c_m \qquad \text{(A.4)}$$

Le délai de propagation $PropDelay_{a,b}$ entre un nœud a et un nœud b est déterminé par la division de la distance $d_{a,b}$ (en mètres) entre les nœuds a et b par la vitesse de propagation s_m du lien du type m connectant ces nœuds a et b; cela peut s'exprimer de la manière suivante:

$$PropDelay_{a,b} = \sum_m \left(\frac{d_{a,b}}{s_m} * v_{a,b,m} \right) \qquad \forall a, \forall b \qquad \text{(A.5)}$$

Le délai de transmission $TransDelay_c$, d'une application du type c placée sur VM_h, est déterminé par la multiplication du nombre μ_{h,c_m} des applications par la quantité de mémoire à transférer ($TransInterMemory_c$) et divisé par le taux de transmission des applications K_{c_m}; cela peut être exprimé comme suit:

$$TransDelay_{h,c_m} = \frac{\mu_{h,c_m} * TransInterMemory_{c_m}}{K_{c_m}} \qquad \forall h, \forall c_m \qquad \text{(A.6)}$$

Le coût d'installation des liens, représenté par C_{links}, englobe les coûts des liens entre VMM_i et $SDSRV_j$, entre $SDSRV_j$ et MR_k, entre $SDSRV_j$ et HR_l. Ce coût C_{links} est exprimé comme suit:

$$C_{links} = \sum_{i \in N_{VMM}} \sum_{j \in N_{SDSRV}} \sum_{m \in T_{VMM,SDSRV}} (\alpha_{i,j,m} * d_{i,j} * v_{i,j,m}) + \sum_{j \in N_{SDSRV}} \sum_{k \in N_{MR}} \sum_{m \in T_{SDSRV,MR}} (\alpha_{j,k,m} * d_{j,k} * v_{j,k,m}) + \sum_{j \in N_{SDSRV}} \sum_{l \in N_{HR}} \sum_{m \in T_{SDSRV,HR}} (\alpha_{j,l,m} * d_{j,l} * v_{j,l,m}) \qquad \text{(A.7)}$$

Le coût total de l'installation des nœuds (VMMs, SDSRVs, MRs, et HRs) est représenté par C_{nodes}; il peut être exprimé de la manière suivante:

$$C_{nodes} = \sum_{i \in N_{VMM}} \sum_{t \in T_{VMM}} \left(\beta_{i,t} * x_{i,t} \right) + \sum_{j \in N_{SDSRV}} \sum_{t \in T_{SDSRV}} \left(\beta_{j,t} * x_{j,t} \right) + \sum_{k \in N_{MR}} \sum_{t \in T_{MR}} \left(\beta_{k,t} * x_{k,t} \right) + \sum_{l \in N_{HR}} \sum_{t \in T_{HR}} \left(\beta_{l,t} * x_{l,t} \right) \quad \text{(A.8)}$$

Chaque VM doit être assignée à un seul VMM; cela peut s'exprimer de la manière suivante:

$$\sum_{i \in N_{VMM}} \sum_{m \in T_{VM,VMM}} v_{h,i,m} = 1 \qquad \forall h \qquad \text{(A.9)}$$

La capacité limitée des flux de chaque nœud y impose les contraintes suivantes:

$$\sum_{h \in N_{VM}} \sum_{c \in (C_{memory} \, U \, C_{hDisk})} \sum_{m \in T_{VM,VMM}} \left(\mu_{h,c} * K_c \; * \; v_{h,i,m} \right) \leq \sum_{t \in T_{VMM}} \left(M_{i,t} * \; x_{i,t} \right) \qquad \forall i \qquad \text{(A.10)}$$

$$\sum_{i \in N_{VMM}} \sum_{c \in (C_{memory} \, U \, C_{hDisk})} f_{i,j,c} \leq \sum_{t \in T_{SDSRV}} \left(M_{j,t} * \; x_{j,t} \right) \qquad \forall j \qquad \text{(A.11)}$$

$$\sum_{j \in N_{SDSRV}} \sum_{c \in C_{memory}} f_{j,k,c} \leq \sum_{t \in T_{MR}} \left(M_{k,t} * \; x_{k,t} \right) \qquad \forall k \qquad \text{(A.12)}$$

$$\sum_{j \in N_{SDSRV}} \sum_{c \in C_{hDisk}} f_{j,l,c} \leq \sum_{t \in T_{HR}} \left(M_{l,t} * \; x_{l,t} \right) \qquad \forall l \qquad \text{(A.13)}$$

Le flux $f_{i,c}$ d'un nœud ou d'un lien doit être égal à la somme de ses flux entrants; cela peut s'exprimer de la manière suivante:

$$f_{i,c} = \sum_{h \in N_{VM}} \sum_{m \in T_{VM,VMM}} \left(\mu_{h,c} * K_c \; * \; v_{h,i,m} \right) \qquad \forall i, \forall c \qquad \text{(A.14)}$$

$$f_{i,j,c} = \sum_{m \in T_{VMM,SDSRV}} \left(f_{i,c} \; * \; v_{i,j,m} \right) \qquad \forall i, \forall j, \forall c \qquad \text{(A.15)}$$

De même, le flux $f_{x,y,c}$ de type c sur un lien connectant un nœud x à un nœud y doit être inférieur de la capacité de ce lien; cela peut s'exprimer de la manière suivante:

$$\sum_{h \in N_{VM}} \sum_{c \in (C_{memory} \, U \, C_{hDisk})} \sum_{m \in T_{VM,VMM}} \left(\mu_{h,c} * K_c \; * \; v_{h,i,m} \right) \leq \sum_{h \in N_{VM}} \sum_{t \in T_{VM,VMM}} \left(\Omega_{h,l,t} * v_{h,l,t} \right) \qquad \forall i \qquad \text{(A.16)}$$

$$\sum_{c \in (C_{memory} \, U \, C_{hDisk})} f_{i,j,c} \; \leq \sum_{t \in T_{VMM,SDSRV}} \left(\Omega_{i,j,t} * v_{i,j,t} \right) \qquad \forall i, \forall j \qquad \text{(A.17)}$$

$$\sum_{c \in C_{memory}} \sum_{t \in T_{SDSRV,MR}} \left(f_{j,k,c} \; * \; v_{j,k,t} \right) \leq \sum_{j \in N_{SDSRV}} \sum_{t \in T_{SDSRV,MR}} \left(\Omega_{j,k,t} * v_{j,k,t} \right) \qquad \forall j, \forall k \qquad \text{(A.18)}$$

$$\sum_{c \in C_{hDisk}} \sum_{m \in T_{SDSRV,HR}} \left(f_{j,l,c} \; * \; v_{j,l,m} \right) \leq \sum_{j \in N_{SDSRV}} \sum_{t \in T_{SDSRV,HR}} \left(\Omega_{j,l,t} * v_{j,l,t} \right) \qquad \forall j, \forall l \qquad \text{(A.19)}$$

De même, chaque VMM doit être assigné au maximum à un seul SDSRV et seulement si ce VMM est placé; chaque SDSRV doit être assigné au maximum à un seul MR et seulement si ce SDSRV est placé; chaque SDSRV doit être assigné au maximum à un seul HR et seulement si ce SDSRV est placé; ces trois contraintes peuvent être exprimées comme suit:

$$\sum_{j \in N_{SDSRV}} \sum_{t \in T_{VMM,SDSRV}} v_{i,j,t} \; \geq \sum_{t \in T_{VMM}} x_{i,t} \qquad \forall i \qquad \text{(A.20)}$$

$$\sum_{k \in N_{MR}} \sum_{t \in T_{SDSRV,MR}} v_{j,k,t} \; \geq \sum_{t \in T_{SDSRV}} x_{j,t} \qquad \forall j \qquad \text{(A.21)}$$

$$\sum_{l \in N_{HR}} \sum_{t \in T_{SDSRV,HR}} v_{j,l,t} \; \geq \sum_{t \in T_{SDSRV}} x_{j,t} \qquad \forall j \qquad \text{(A.22)}$$

Chaque emplacement des nœuds doit avoir au maximum un nœud placé; cela peut s'exprimer de la manière suivante:

$$\sum_{t \in T_{VMM}} x_{i,t} \leq 1 \qquad \forall i \qquad \text{(A.23)}$$

$$\sum_{t \in T_{SDSRV}} x_{j,t} \leq 1 \qquad \forall j \qquad \text{(A.24)}$$

$$\sum_{t \in T_{MR}} x_{k,t} \leq 1 \qquad \forall k \qquad \text{(A.25)}$$

$$\sum_{t \in T_{HR}} x_{l,t} \leq 1 \qquad \forall l \qquad \text{(A.26)}$$

Le flux de chaque nœud est conservé et il est égal au flux sortant de ce nœud; ces conservations de flux peuvent être exprimées de la manière suivante:

$$\sum_{h \in N_{VM}} \sum_{m \in T_{VM,VMM}} \left(\mu_{h,c} * K_c * v_{h,i,m} \right) = \sum_{j \in N_{SDSRV}} f_{i,j,c} \qquad \forall i, \forall c \qquad \text{(A.27)}$$

$$\sum_{i \in N_{VMM}} \sum_{m \in T_{VMM,SDSRV}} f_{i,j,c_m} = \sum_{k \in N_{MR}} f_{j,k,c_m} \qquad \forall j, \forall c_m \qquad \text{(A.28)}$$

$$\sum_{i \in N_{VMM}} \sum_{m \in T_{VMM,SDSRV}} f_{i,j,c_h} = \sum_{l \in N_{HR}} f_{j,l,c_h} \qquad \forall j, \forall c_h \qquad \text{(A.29)}$$

En conclusion, le problème GPV d'assignation (et de placement) des VMs interdépendantes consiste à minimiser la fonction objectif f (A. 1) sous les contraintes (A.2) – (A.29); comme décrit auparavant, le coût total f (A. 1) comprend deux composantes, c.à.d., le coût de l'installation des nœuds (VMs, VMMs, SDSRVs, MRs, HRs), et le coût des liens entre ces nœuds.

ANNEXE B - RÉSULTATS ADDITIONNELS DE PLACEMENT DE MACHINES VIRTUELLES INTERDÉPENDANTES

Cette annexe présente des résultats additionnels concernant le problème de placement des machines virtuelles interdépendantes (GPV); ce problème est résolu en utilisant le modèle mathématique du quatrième chapitre détaillé dans l'annexe A; ce modèle mathématique est approximé par l'heuristique de recherche taboue (TS) présentée dans le quatrième chapitre; les résultats de TS montrent qu'elle permet d'obtenir des solutions quasi-optimales pour ce problème NP-difficile de placement de VMs interdépendantes (GPV).

Afin d'avoir des tests additionnels pour le placement de VMs interdépendantes, nous implémentons notre modèle mathématique avec la méthode de programmation en nombres entiers mixte (MIP) en utilisant le solveur CPLEX [33]; de même, nous utilisons le langage C# (Framework .Net 4.5) pour développer l'heuristique TS; nous testons notre MIP et TS avec Windows installé sur PC ayant Intel (R) Core I7 4GHz et 16GB de mémoire; les paramètres de TS et du MIP-R exécuté par CPLEX sont définis au Tableau B.1 et Figure B.1 respectivement; les différents types des nœuds et des liens sont définis au Tableau B.2.

Lors de l'utilisation du MIP avec CPLEX, nous obtenons des erreurs de dépassement de mémoire dues aux contraintes des délais d'interdépendance (A.2) - (A.6); pour cette raison, nous utilisons MIP-R qui est une version relaxée du MIP; MIP-R nous permet d'avoir des bornes inferieures pour des solutions de GPV, et il comporte toutes les contraintes du MIP à l'exception des contraintes (A.2) - (A.6);

L'interdépendance entre les services est choisie aléatoirement selon la "distribution Bernoulli" (avec la probabilité $p = 1/3$); ces services partagent des données avec un taux choisi uniformément dans l'intervalle [0.1,1] par seconde; nous choisissons nos nœuds à partir d'ensembles initiaux prédéfinis; ces ensembles initiaux contiennent 40 VMs, 30 VMMs, 10 SDSRVs, 10 MRs, et 10 HRs; nos tests sont exécutés avec quarante-huit ensembles (générés aléatoirement) pour notre problème GPV; nous choisissons aléatoirement les emplacements des VMs et les emplacements potentiels des VMMs, SDSRVs, MRs, et HRs; ainsi, nous obtenons différents ensembles de tests pour notre problème GPV NP-difficile; chaque exécution de l'heuristique de recherche taboue (TS) est répétée 100 fois pour déduire la moyenne de ses résultats; l'analyse et l'interprétation des résultats sont basées sur le coût total (en dollars) des

solutions trouvées et de leur délai CPU (en secondes). Les compositions de nos scenarios de tests sont décrites au Tableau B.3; ces scenarios sont exécutés à la fois par TS et MIP-R; les résultats sont présentés au Tableau B.4, Tableau B.5, Figure B.2, et Figure B.3.

Les résultats des différents scenarios (voir Tableau B.4 et Tableau B.5) montrent clairement que notre TS trouvent rapidement des solutions proches des bornes inferieurs obtenues via MIP-R pour le problème GPV NP-difficile; les résultats obtenus montrent que l'heuristique TS produit des solutions qui sont en moyenne à 1,57% et au maximum à 4,14% des limites inférieures des bornes inferieures des solutions optimales obtenues avec MIP-R; notre TS obtient des résultats dans des délais acceptables avec un délai moyen de 60 secondes; les différences entre les délais de TS et MIP-R ne dépassent pas 214 secondes (voir Tableau B.5); comme résultat, notre TS est excellente puisque le problème de la planification des réseaux virtuels est un problème NP-difficile à résoudre surtout pour des grands ensembles de VMs; nous appliquons notre TS sur des ensembles plus grands du problème GPV (avec des milliers de machines virtuelles) et nous présentons les résultats au Tableau B.6 et Tableau B.7.

Tableau B.1: Notation et définitions

Paramètre	Valeur
$max\ Iterations$:	10.
$max\ Neighbours$:	Maximum des nombres N_{VMM}, N_{SDSRV}, N_{MR}, N_{HR}, T_{VMM},T_{SDSRV},T_{MR}, T_{HR}, $T_{VM,VMM}$,$T_{VMM,SDSRV}$, $T_{SDSRV,MR}$, and $T_{SDSRV,HR}$.

```
#CHANGE THE SOLVER (optional)
option solver cplex;
#option solver cplex1251;
#option cplex_options 'timing 1';
#option log_file 'mphapp2.out';
option log_file '.\multi';
#option cplex_options 'mipdisplay 2 mipinterval 1000000';
option solution_precision 10;
#option solver_msg 0;
option version;
option presolve 1;
#option cplex_options 'iisfind 1';
#option solver bonmin;
#option cplex_options 'mipdisplay=2';
option show_stats 1;
option cplex_options 'mipdisplay 2 mipinterval 1';
```

Figure B.1: Paramètres de CPLEX pour MIP-R

Tableau B.2: Différents types des différents nœuds et liens de notre architecture de GPV

Type du nœud/lien	Capacité (Mbps)	Coût ($)
VMM – Type A	120	20000
VMM – Type B	240	30000
VMM – Type C	480	50000
SDSRV – Type A	2000	50000
SDSRV – Type B	5000	90000
SDSRV – Type C	10000	120000
MR – Type A	20000	200000
MR – Type B	40000	350000
MR – Type C	60000	500000
HR – Type A	20,000	40000
HR – Type B	40000	60000
Lien VM vers VMM – Type A	15	600
Lien VMM vers SDSRV - Type A	155	1500
Lien VMM vers SDSRV - Type B	622	4000
Lien SDSRV vers MR - Type A	1000	2500
Lien SDSRV vers HR - Type A	1000	4000

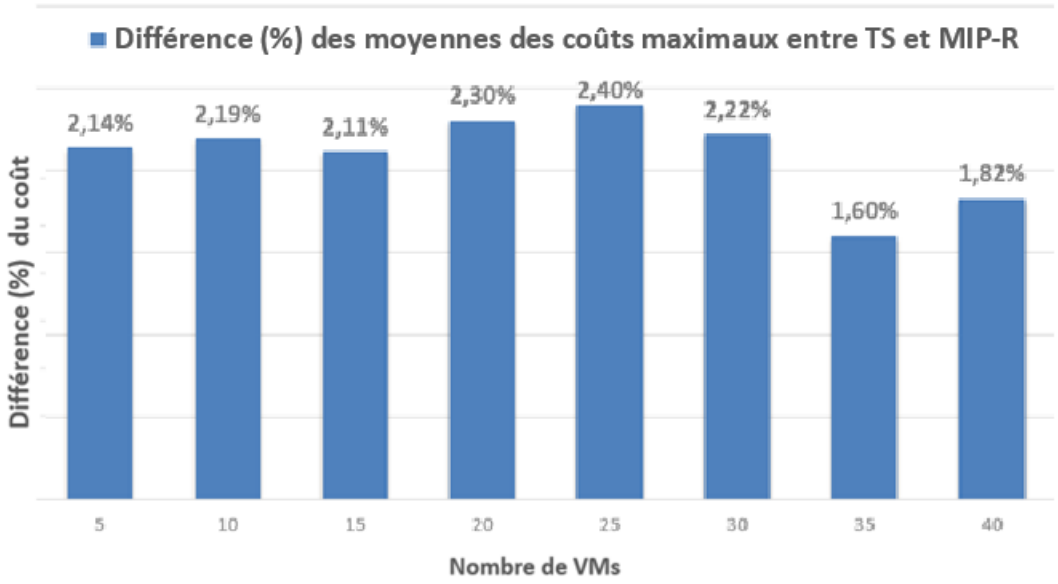

Figure B.2: Écart moyen (en %) entre le coût de TS et MIP-R

Tableau B.3: Description de nos quarante-huit scénarios de test du problème GPV

#Test	|VMs|	|VMMs|	|SDSRVs|	|MRs|	|HRs|	#Test	|VMs|	|VMMs|	|SDSRVs|	|MRs|	|HRs|
1	5	10	5	5	5	**25**	25	10	5	5	5
2	5	10	10	10	10	**26**	25	10	10	10	10
3	5	20	5	5	5	**27**	25	20	5	5	5
4	5	20	10	10	10	**28**	25	20	10	10	10
5	5	30	5	5	5	**29**	25	30	5	5	5
6	5	30	10	10	10	**30**	25	30	10	10	10
7	10	10	5	5	5	**31**	30	10	5	5	5
8	10	10	10	10	10	**32**	30	10	10	10	10
9	10	20	5	5	5	**33**	30	20	5	5	5
10	10	20	10	10	10	**34**	30	20	10	10	10
11	10	30	5	5	5	**35**	30	30	5	5	5
12	10	30	10	10	10	**36**	30	30	10	10	10
13	15	10	5	5	5	**37**	35	10	5	5	5
14	15	10	10	10	10	**38**	35	10	10	10	10
15	15	20	5	5	5	**39**	35	20	5	5	5
16	15	20	10	10	10	**40**	35	20	10	10	10
17	15	30	5	5	5	**41**	35	30	5	5	5
18	15	30	10	10	10	**42**	35	30	10	10	10
19	20	10	5	5	5	**43**	40	10	5	5	5
20	20	10	10	10	10	**44**	40	10	10	10	10
21	20	20	5	5	5	**45**	40	20	5	5	5
22	20	20	10	10	10	**46**	40	20	10	10	10
23	20	30	5	5	5	**47**	40	30	5	5	5
24	20	30	10	10	10	**48**	40	30	10	10	10

Tableau B.4: Comparaison des coûts obtenus par TS et MIP-R des différents scénarios du GPV

#Test	TS			MIP-R (CPLEX)	Écart (%) TS vs MIP-R		
	Coût ($)			Coût ($)	Coût		
	Min	Moy	Max		Min	Moy	Max
1	2818638	2848638	2868638	2808638	0.36%	1.42%	2.14%
2	2133511	2153511	2173511	2113511	0.95%	1.89%	2.84%
3	2805844	2805844	2805844	2765844	1.45%	1.45%	1.45%
4	1453001	1456001	1461001	1451001	0.14%	0.34%	0.69%
5	2577043	2577043	2577043	2537043	1.58%	1.58%	1.58%
6	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
7	3116259	3116259	3116259	3076259	1.30%	1.30%	1.30%
8	1461001	1491001	1511001	1451001	0.69%	2.76%	4.14%
9	2647061	2650061	2655061	2645061	0.08%	0.19%	0.38%
10	1491001	1491001	1491001	1451001	2.76%	2.76%	2.76%
11	2232870	2232870	2232870	2192870	1.82%	1.82%	1.82%
12	1491001	1491001	1491001	1451001	2.76%	2.76%	2.76%
13	2502456	2532456	2552456	2492456	0.40%	1.60%	2.41%
14	1757008	1760008	1765008	1755008	0.11%	0.28%	0.57%
15	1757008	1760008	1765008	1755008	0.11%	0.28%	0.57%
16	1775008	1795008	1815008	1755008	1.14%	2.28%	3.42%
17	2577043	2577043	2577043	2537043	1.58%	1.58%	1.58%
18	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
19	3101973	3131973	3151973	3091973	0.32%	1.29%	1.94%
20	1775008	1795008	1815008	1755008	1.14%	2.28%	3.42%
21	3072842	3075842	3080842	3070842	0.07%	0.16%	0.33%
22	1795008	1795008	1795008	1755008	2.28%	2.28%	2.28%
23	2391772	2391772	2391772	2351772	1.70%	1.70%	1.70%
24	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
25	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
26	1453001	1456001	1461001	1451001	0.14%	0.34%	0.69%
27	1512132	1512132	1512132	1472132	2.72%	2.72%	2.72%
28	2030217	2030217	2030217	1990217	2.01%	2.01%	2.01%
29	1461001	1491001	1511001	1451001	0.69%	2.76%	4.14%
30	1453001	1456001	1461001	1451001	0.14%	0.34%	0.69%
31	2598253	2601253	2606253	2596253	0.08%	0.19%	0.39%
32	1992217	1995217	2000217	1990217	0.10%	0.25%	0.50%
33	3277737	3297737	3317737	3257737	0.61%	1.23%	1.84%
34	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
35	2617932	2637932	2657932	2597932	0.77%	1.54%	2.31%
36	1471001	1491001	1511001	1451001	1.38%	2.76%	4.14%
37	2466124	2469124	2474124	2464124	0.08%	0.20%	0.41%
38	1757008	1760008	1765008	1755008	0.11%	0.28%	0.57%
39	2877241	2880241	2885241	2875241	0.07%	0.17%	0.35%
40	1461001	1491001	1511001	1451001	0.69%	2.76%	4.14%
41	2942760	2942760	2942760	2902760	1.38%	1.38%	1.38%
42	1491001	1491001	1491001	1451001	2.76%	2.76%	2.76%
43	2670498	2670498	2670498	2630498	1.52%	1.52%	1.52%
44	1757008	1760008	1765008	1755008	0.11%	0.28%	0.57%
45	2202870	2232870	2252870	2192870	0.46%	1.82%	2.74%
46	1757008	1760008	1765008	1755008	0.11%	0.28%	0.57%
47	2871046	2871046	2871046	2831046	1.41%	1.41%	1.41%
48	1461001	1491001	1511001	1451001	0.69%	2.76%	4.14%
Statistiques					**0.07%**	**1.57%**	**4.14%**

Tableau B.5: Comparaison des délais de TS et MIP-R des différents scénarios du GPV

#Test	TS - Délai (sec)			MIP-R	Écart du délai de TS vs MIP-R (sec)		
	Min	Moy	Max	Délai (sec)	Min	Moy	Max
1	3,586	3,592	3,602	0.23 (110.81 ticks)	3,36	3,36	3,37
2	14,881	14,94	15,021	1.92 (1807.34 ticks)	12,96	13,02	13,10
3	6,158	6,197333	6,266	1.70 (1367.55 ticks)	4,46	4,50	4,57
4	22,435	22,473	22,531	2.00 (1681.24 ticks)	20,44	20,47	20,53
5	22,435	22,473	22,531	2.56 (2256.77 ticks)	19,88	19,91	19,97
6	9,311	9,320667	9,33	1.31 (1025.68 ticks)	8,00	8,01	8,02
7	31,479	31,55833	31,612	0.55 (438.54 ticks)	30,93	31,01	31,06
8	5,896	5,939333	6,019	0.59 (384.07 ticks)	5,31	5,35	5,43
9	20,908	20,96967	21,003	1.72 (1226.33 ticks)	19,19	19,25	19,28
10	9,539	9,594333	9,68	0.92 (665.78 ticks)	8,62	8,67	8,76
11	30,883	31,01667	31,115	0.86 (583.48 ticks)	30,02	30,16	30,26
12	13,953	14,00767	14,08	6.57 (5194.06 ticks)	7,38	7,44	7,51
13	42,488	42,50567	42,534	0.86 (752.97 ticks)	41,63	41,65	41,67
14	9,69	9,733667	9,809	1.93 (1739.03 ticks)	7,76	7,80	7,88
15	30,665	30,669	30,674	3.40 (2629.32 ticks)	27,27	27,27	27,27
16	14,998	15,01333	15,042	6.86 (5418.52 ticks)	8,14	8,15	8,18
17	43,993	44,00233	44,01	1.89 (1464.89 ticks)	42,10	42,11	42,12
18	21,262	21,354	21,494	2.50 (1978.52 ticks)	18,76	18,85	18,99
19	59,564	59,64367	59,799	0.27 (215.57 ticks)	59,29	59,37	59,53
20	15,023	15,08467	15,117	2.29 (1844.05 ticks)	12,73	12,79	12,83
21	44,659	44,75133	44,83	2.09 (1582.50 ticks)	42,57	42,66	42,74
22	22,728	22,83767	22,964	7.27 (5297.06 ticks)	15,46	15,57	15,69
23	62,972	63,04467	63,082	2.34 (1738.47 ticks)	60,63	60,70	60,74
24	31,765	31,775	31,795	3.37 (2649.23 ticks)	28,40	28,41	28,43
25	83,942	83,99133	84,035	0.56 (364.34 ticks)	83,38	83,43	83,48
26	22,231	22,339	22,433	1.34 (986.04 ticks)	20,89	21,00	21,09
27	63,662	63,71167	63,749	1.03 (674.91 ticks)	62,63	62,68	62,72
28	33,516	33,73967	34,067	8.95 (6633.24 ticks)	24,57	24,79	25,12
29	89,286	89,359	89,445	4.13 (3273.92 ticks)	85,16	85,23	85,32
30	46,221	46,29867	46,419	4.93 (3602.62 ticks)	41,29	41,37	41,49
31	117,854	117,902	117,992	0.36 (282.21 ticks)	117,49	117,54	117,63
32	31,991	32,095	32,204	3.29 (2456.31 ticks)	28,70	28,81	28,91
33	88,492	88,93433	89,168	3.43 (2753.40 ticks)	85,06	85,50	85,74
34	46,845	54,54267	69,688	2.42 (1821.11 ticks)	44,43	52,12	67,27
35	120,392	130,3717	149,079	6.40 (4679.27 ticks)	113,99	123,97	142,68
36	62,782	63,526	64,348	23.49(16076.4ticks)	39,29	40,04	40,86
37	154,869	160,394	170,744	1.09 (849.88 ticks)	153,78	159,30	169,65
38	43,133	43,24867	43,48	3.63 (2856.70 ticks)	39,50	39,62	39,85
39	118,375	118,8223	119,57	3.03 (2349.33 ticks)	115,35	115,79	116,54
40	63,229	63,492	63,625	12.61 (8851.91ticks)	50,62	50,88	51,02
41	160,952	165,555	174,192	8.52 (7162.34 ticks)	152,43	157,04	165,67
42	84,836	85,079	85,272	6.33 (4839.75 ticks)	78,51	78,75	78,94
43	208,778	208,9563	209,178	2.56 (2159.91 ticks)	206,22	206,40	206,62
44	57,092	60,07167	66,031	8.97 (6999.10 ticks)	48,12	51,10	57,06
45	154,282	158,5853	164,346	2.03 (1649.71 ticks)	152,25	156,56	162,32
46	83,957	89,15533	99,516	9.39 (7261.42 ticks)	74,57	79,77	90,13
47	209,143	213,6623	221,218	7.28 (6553.04 ticks)	201,86	206,38	213,94
48	110,515	110,9037	111,109	7.22 (5229.59 ticks)	103,30	103,68	103,89
Statistiques	**3,59**	**60,36**	**222,22**		**3,36**	**56,42**	**213,94**

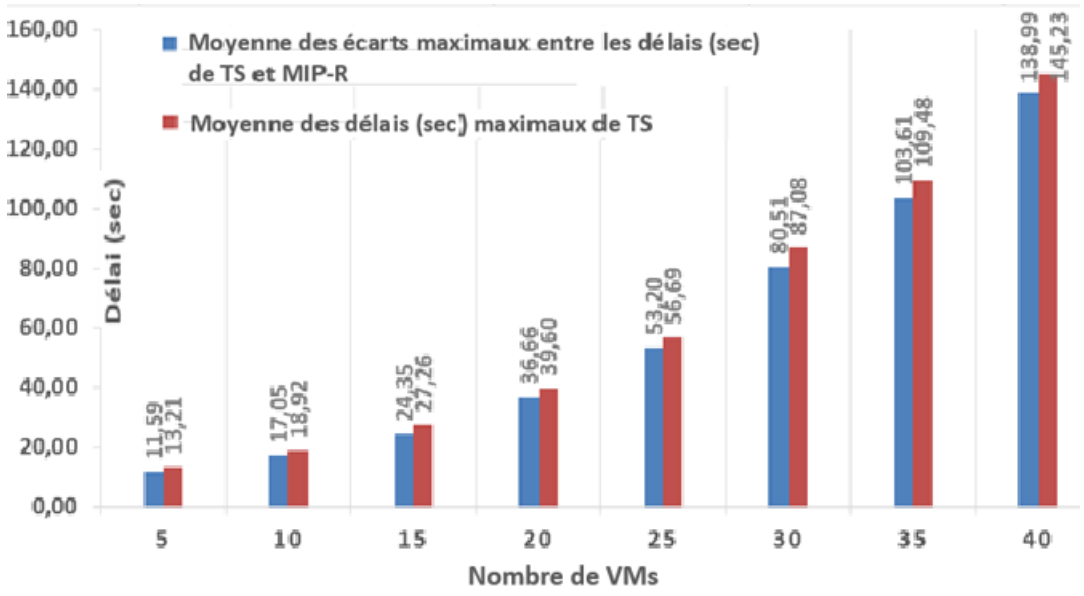

Figure B.3: Moyennes des maximums des temps CPU de TS et MIP-R

Nous essayons de résoudre les grands ensembles de notre problème GPV (avec des milliers de machines virtuelles) avec MIP-R, mais nous obtenons des erreurs de "dépassement de mémoire"; notre TS (validée par les résultats aux Figure B.2 et Figure B.3) obtient des résultats avec des délais acceptables (voir Tableau B.6, Tableau B.7, Figure B.4, et Figure B.5); ces tableaux montrent les résultats de nos nouveaux scénarios (de milliers de VMs) de notre problème de planification GPV; chaque nouveau scénario est une agrégation de 1000 copies d'un des scénarios détaillés auparavant au Tableau B.3. Les résultats de notre TS avec les nouveaux scénarios sont meilleurs que ceux de FFD; nous utilisons les solutions obtenues par FFD comme base de comparaison car il est difficile d'avoir des bornes inférieures avec MIP-R pour des ensembles de GPV avec des milliers de machines virtuelles; MIP-R rencontre des problèmes de "dépassement de mémoire" lorsqu'il est exécuté sur les nouveaux grands scénarios qui sont l'agrégation de 1000 copies des scénarios du Tableau B.3; les résultats de notre TS sont généralement meilleurs que ceux de FFD et ils sont obtenus dans un laps de temps raisonnable; les résultats obtenus montrent que l'heuristique TS produit des solutions meilleures en moyenne de 71,13%, au minimum de 67,32%, et au maximum de 83,70% des solutions FFD (voir Tableau B.6); notre TS obtient des résultats dans des délais acceptables, qui ont un écart moyen de 29305 secondes et maximal de 109005 secondes des délais

de FFD (voir Tableau B.7); comme résultat, notre TS est excellente puisque le problème de la planification des réseaux virtuels est un problème NP-difficile et difficile à résoudre avec MIP-R surtout pour des grands ensembles de VMs interdépendantes.

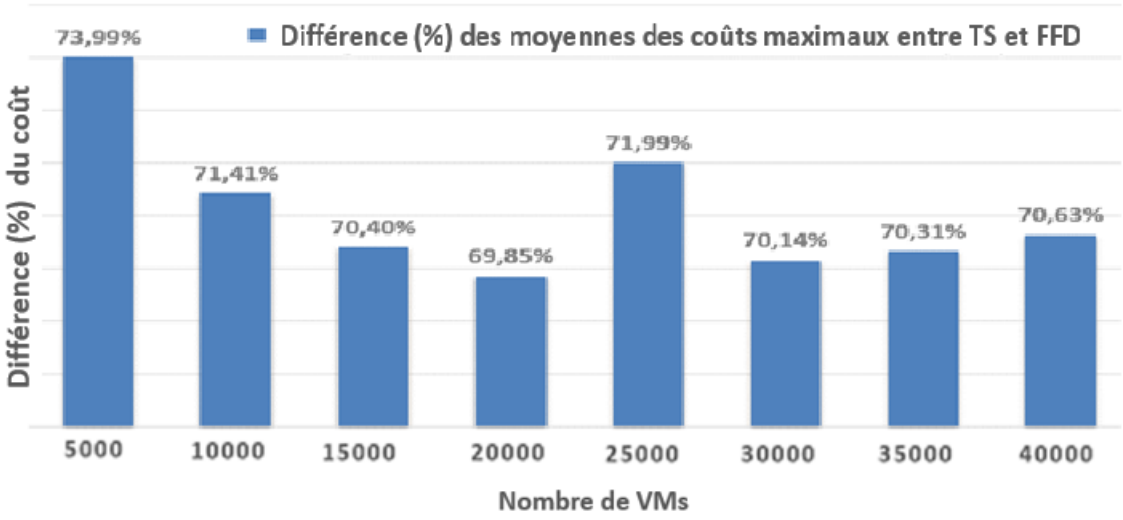

Figure B.4: Écart de coût (en %) entre les solutions de TS et FFD

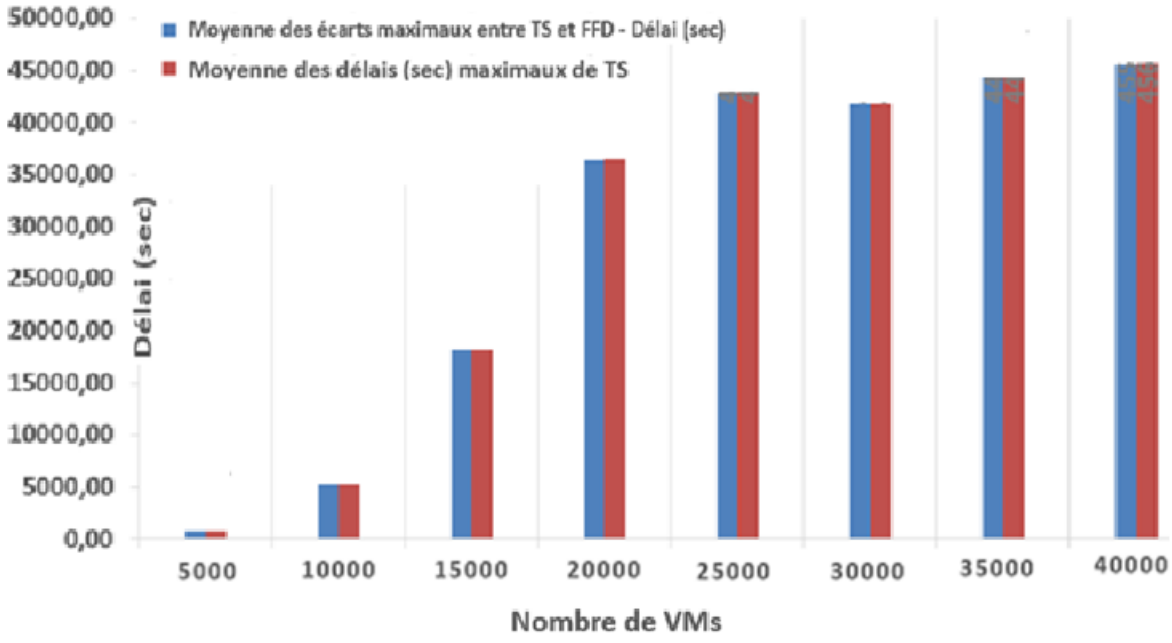

Figure B.5: Moyennes des délais CPU maximaux de TS

Tableau B.6: Coûts de TS pour les nouveaux scénarios du GPV (avec grands ensembles de VMs)

#Test	TS - Coût ($)			FFD - Coût ($)	Écart (en %) des coûts de TS vs FFD		
	Min	Moy	Max		Max	Moy	Min
1 avec 5000 VMs	6602731	6632731	6652731.2	22391982.34	70.51%	70.38%	70.29%
2 avec 5000 VMs	6230995	6250995	6270994.8	25847178.84	75.89%	75.82%	75.74%
3 avec 5000 VMs	7114178	7114178	7114178.17	36981801.68	80.76%	80.76%	80.76%
4 avec 5000 VMs	8522704	8525704	8530703.64	30722033.73	72.26%	72.25%	72.23%
5 avec 5000 VMs	7270633	7270633	7270632.74	25980593.16	72.02%	72.02%	72.02%
6 avec 5000 VMs	5742939	5762939	5782939.19	21348443.93	73.10%	73.01%	72.91%
8 avec 10000 VMs	11967498	11967498	11967497.8	43166687.93	72.28%	72.28%	72.28%
9 avec 10000 VMs	10194771	10224771	10244771.5	31193812.39	67.32%	67.22%	67.16%
9 avec 10000 VMs	14905544	14908544	14913544.4	51225603.49	70.90%	70.90%	70.89%
10 avec 10000 VMs	11133811	11133811	11133810.9	40495627.28	72.51%	72.51%	72.51%
11 avec 10000 VMs	16001758	16001758	16001757.5	56601025.6	71.73%	71.73%	71.73%
12 avec 10000 VMs	13961946	13961946	13961945.6	55377173.19	74.79%	74.79%	74.79%
13 avec 15000 VMs	14549944	14579944	14599944.5	48507910.75	70.01%	69.94%	69.90%
14 avec 15000 VMs	9682631	9685631	9690630.92	34414434.06	71.86%	71.86%	71.84%
15 avec 15000 VMs	15477542	15480542	15485541.6	51049942	69.68%	69.68%	69.67%
16 avec 15000 VMs	16494367	16514367	16534366.5	56637294.07	70.88%	70.84%	70.81%
17 avec 15000 VMs	21127189	21127189	21127189.1	70475761.8	70.02%	70.02%	70.02%
18 avec 15000 VMs	19955397	19975397	19995397.1	67020385.84	70.22%	70.20%	70.17%
19 avec 20000 VMs	10486916	10516916	10536915.9	35883801.22	70.78%	70.69%	70.64%
20 avec 20000 VMs	11018849	11038849	11058849.2	39541444.22	72.13%	72.08%	72.03%
21 avec 20000 VMs	20205061	20208061	20213061.4	66616798.3	69.67%	69.67%	69.66%
22 avec 20000 VMs	19867210	19867210	19867210	60818837.78	67.33%	67.33%	67.33%
23 avec 20000 VMs	22584603	22584603	22584602.6	74848002.32	69.83%	69.83%	69.83%
24 avec 20000 VMs	24149628	24169628	24189627.6	79603077.18	69.66%	69.64%	69.61%
25 avec 25000 VMs	9450196	9470196	9490195.62	58237714.84	83.77%	83.74%	83.70%
26 avec 25000 VMs	10533958	10536958	10541958.3	35235345.63	70.10%	70.10%	70.08%
27 avec 25000 VMs	20748154	20748154	20748154.2	69388657.01	70.10%	70.10%	70.10%
28 avec 25000 VMs	20543091	20543091	20543091.1	68773467.75	70.13%	70.13%	70.13%
29 avec 25000 VMs	28674883	28704883	28724882.7	92509618.25	69.00%	68.97%	68.95%
30 avec 25000 VMs	28666883	28669883	28674882.7	92509618.25	69.01%	69.01%	69.00%
31 avec 30000 VMs	7451187	7454187	7459186.85	27720101.83	73.12%	73.11%	73.09%
32 avec 30000 VMs	8333911	8336911	8341911.24	28320724.55	70.57%	70.56%	70.54%
33 avec 30000 VMs	22610292	22630292	22650292	75035070.46	69.87%	69.84%	69.81%
34 avec 30000 VMs	18662648	18682648	18702648.2	60388859.2	69.10%	69.06%	69.03%
35 avec 30000 VMs	33507533	33527533	33547533.5	109742742.7	69.47%	69.45%	69.43%
36 avec 30000 VMs	32018386	32038386	32058386.3	103259353.3	68.99%	68.97%	68.95%
37 avec 35000 VMs	11257254	11260254	11265253.7	40338592.69	72.09%	72.09%	72.07%
38 avec 35000 VMs	10141219	10144219	10149219.3	36990489.35	72.58%	72.58%	72.56%
39 avec 35000 VMs	21764721	21767721	21772720.7	71860993.64	69.71%	69.71%	69.70%
40 avec 35000 VMs	21666868	21696868	21716867.5	71491272.09	69.69%	69.65%	69.62%
41 avec 35000 VMs	32365327	32365327	32365326.9	104240175	68.95%	68.95%	68.95%
42 avec 35000 VMs	32038386	32038386	32038386.3	103259353.3	68.97%	68.97%	68.97%
43 avec 40000 VMs	10403729	10403729	10403728.8	37489070.9	72.25%	72.25%	72.25%
44 avec 40000 VMs	8829190	8832190	8837190.15	33717829.95	73.81%	73.81%	73.79%
45 avec 40000 VMs	19924098	19954098	19974097.8	66358245.34	69.97%	69.93%	69.90%
46 avec 40000 VMs	22041509	22044509	22049508.5	73382719.99	69.96%	69.96%	69.95%
47 avec 40000 VMs	32920898	32920898	32920897.5	105906887	68.92%	68.92%	68.92%
48 avec 40000 VMs	32008386	32038386	32058386.3	103259353.3	69.00%	68.97%	68.95%
Statistiques	**5742939**	**17131532**	**33547533**		**83,70%**	**71,13%**	**67,32%**

Tableau B.7: Délais de TS pour les nouveaux scénarios du GPV (avec grands ensembles de VMs)

#Test	TS - Délai (sec)			FFD - Délai (sec)	Écart des délais de TS vs FFD (sec)		
	Min	Moy	Max		Min	Moy	Max
1 avec 5000 VMs	390.056	400.056	460.056	3.466	386.59	396.59	456.59
2 avec 5000 VMs	534.544	554.544	614.544	3.389	531.16	551.16	611.16
3 avec 5000 VMs	623.236	663.236	703.236	3.448	619.79	659.79	699.79
4 avec 5000 VMs	838.787	840.787	850.787	3.449	835.34	837.34	847.34
5 avec 5000 VMs	833.931	873.931	913.931	3.417	830.51	870.51	910.51
6 avec 5000 VMs	882.509	902.509	962.509	3.532	878.98	898.98	958.98
8 avec 10000 VMs	1691.123	1691.123	1691.123	7.987	1683.14	1683.14	1683.14
9 avec 10000 VMs	3889.163	3899.163	3959.163	13.785	3875.38	3885.38	3945.38
9 avec 10000 VMs	4306.213	4308.213	4318.213	14.231	4291.98	4293.98	4303.98
10 avec 10000 VMs	5357.032	5397.032	5437.032	14.358	5342.67	5382.67	5422.67
11 avec 10000 VMs	5674.497	5714.497	5754.497	13.977	5660.52	5700.52	5740.52
12 avec 10000 VMs	7045.003	7085.003	7125.003	14.354	7030.65	7070.65	7110.65
13 avec 15000 VMs	1959.545	1969.545	2029.545	9.65	1949.90	1959.90	2019.90
14 avec 15000 VMs	2721.294	2723.294	2733.294	9.738	2711.56	2713.56	2723.56
15 avec 15000 VMs	35253.37	35255.37	35265.374	34.478	35218.90	35220.90	35230.90
16 avec 15000 VMs	19257.61	19277.61	19337.606	35.006	19222.60	19242.60	19302.60
17 avec 15000 VMs	24135.74	24175.74	24215.739	36.07	24099.67	24139.67	24179.67
18 avec 15000 VMs	25791.33	25811.33	25871.329	35.193	25756.14	25776.14	25836.14
19 avec 20000 VMs	2138.262	2148.262	2208.262	10.531	2127.73	2137.73	2197.73
20 avec 20000 VMs	2973.009	2993.009	3053.009	10.779	2962.23	2982.23	3042.23
21 avec 20000 VMs	37642.14	37644.14	37654.139	64.523	37577.62	37579.62	37589.62
22 avec 20000 VMs	49922.55	49962.55	50002.553	70.977	49851.58	49891.58	49931.58
23 avec 20000 VMs	55132.34	55172.34	55212.337	70.19	55062.15	55102.15	55142.15
24 avec 20000 VMs	70825.54	70845.54	70905.538	71.959	70753.58	70773.58	70833.58
25 avec 25000 VMs	2716.158	2736.158	2796.158	12.485	2703.67	2723.67	2783.67
26 avec 25000 VMs	3409.798	3411.798	3421.798	12.766	3397.03	3399.03	3409.03
27 avec 25000 VMs	22650.11	22690.11	22730.113	37.062	22613.05	22653.05	22693.05
28 avec 25000 VMs	25058.56	25098.56	25138.558	34.747	25023.81	25063.81	25103.81
29 avec 25000 VMs	101574.6	101584.6	101644.647	99.843	101474.80	101484.80	101544.80
30 avec 25000 VMs	101582.6	101584.6	101594.647	99.843	101482.80	101484.80	101494.80
31 avec 30000 VMs	2331.106	2333.106	2343.106	11.396	2319.71	2321.71	2331.71
32 avec 30000 VMs	3710.145	3712.145	3722.145	22.708	3687.44	3689.44	3699.44
33 avec 30000 VMs	22883.78	22903.78	22963.776	40.228	22843.55	22863.55	22923.55
34 avec 30000 VMs	26146.22	26166.22	26226.219	34.91	26111.31	26131.31	26191.31
35 avec 30000 VMs	91997.35	92017.35	92077.348	79.187	91918.16	91938.16	91998.16
36 avec 30000 VMs	103880.4	103900.4	103960.374	352.024	103528.35	103548.35	103608.35
37 avec 35000 VMs	3110.676	3112.676	3122.676	15.127	3095.55	3097.55	3107.55
38 avec 35000 VMs	4408.086	4410.086	4420.086	15.461	4392.63	4394.63	4404.63
39 avec 35000 VMs	24942.69	24944.69	24954.688	41.432	24901.26	24903.26	24913.26
40 avec 35000 VMs	28293.92	28303.92	28363.922	34.909	28259.01	28269.01	28329.01
41 avec 35000 VMs	97351.19	97391.19	97431.191	81.852	97269.34	97309.34	97349.34
42 avec 35000 VMs	108325.7	108365.7	108405.713	77.593	108248.12	108288.12	108328.12
43 avec 40000 VMs	3412.754	3452.754	3492.754	17.298	3395.46	3435.46	3475.46
44 avec 40000 VMs	4582.955	4584.955	4594.955	16.904	4566.05	4568.05	4578.05
45 avec 40000 VMs	24829.83	24839.83	24899.832	40.304	24789.53	24799.53	24859.53
46 avec 40000 VMs	32262.62	32264.62	32274.617	47.712	32214.91	32216.91	32226.91
47 avec 40000 VMs	99385.58	99425.58	99465.581	87.853	99297.73	99337.73	99377.73
48 avec 40000 VMs	109007	109017	109077.008	72.078	108934.93	108944.93	109004.93
Statistiques	**390,06**	**29345,02**	**109077.01**		**386.59**	**29304.51**	**109004.93**

Tableau B.8: Résumé des statistiques clés des résultats de notre TS

	Différence entre TS et MIP-R		Différence entre TS et FFD	
	Coût (en %)	**Délai (sec)**	**Coût (en %)**	**Délai (sec)**
Moyenne	1,57%	56,42	71,13%	29304,51
Écart-type	0,99%	52,89	2,94%	36315,49
Taille de l'échantillon	48	48	48	48
Coefficient de confiance	1,96	1,96	1,96	1,96
Marge d'erreur	0,28%	14,96	0,83%	10273,71
Borne supérieure	1,72%	64,05	71,56%	34546,20
Borne inférieure	1,43%	48,79	70,71%	24062,82
Max	2,76%	206,40	83,74%	108944,93
Min	0,16%	3,36	67,22%	396,59
Marge (Range)	2,60%	203,04	16,52%	108548,34

En conclusion, cette annexe présente une recherche taboue (TS) pour résoudre le problème GPV de la planification des machines virtuelles interdépendantes dans un centre de données; les résultats de TS indiquent qu'il est possible de trouver des solutions quasi-optimales (voir Tableau B.8) pour GPV où les machines virtuelles interdépendantes peuvent échanger efficacement leurs données.

ANNEXE C - RÉSULTATS ADDITIONNELS POUR LE PROBLÈME DE MIGRATION EN TEMPS RÉEL DES VMS INTERDÉPENDANTES PRENANT EN CONSIDÉRATION LES ENJEUX DES CONTRATS SLAS

Cette annexe présente des résultats additionnels concernant la modélisation mathématique des enjeux de qualité de service de la migration en temps réel des machines virtuelles interdépendantes. Cette modélisation mathématique est décrite dans le cinquième chapitre; cette annexe présente donc des résultats additionnels de cette modélisation mathématiques et de son heuristique d'approximation; ces résultats montrent qu'il est possible de réduire au minimum les temps d'arrêt des migrations en temps réel et les délais de transmission des machines virtuelles interdépendantes; ces résultats sont réalisés via des simulations qui montrent qu'il est possible de respecter des contrats SLAs dans un centre de données et de maintenir la qualité de service des applications interdépendantes tandis que leur VMs hôtes effectuent des migrations en temps réel.

Les résultats de cette annexe sont basées sur le modèle mathématique MIP (du cinquième chapitre) construit des équations mathématiques (5.1) - (5.22); ce modèle résout le problème de migration en temps réel des VMs interdépendantes prenant en considération les enjeux des contrats SLAs (SARTLM); ce problème SARTLM est NP-difficile puisque le problème "bin-packing" [9] peut être réduite à SARTLM; pour cette raison, nous concevons notre heuristique TS et nous l'utilisons pour résoudre notre problème SARTLM dans des délais acceptables pour des grands ensembles de VMs; autres heuristiques pourraient aussi trouver des solutions quasi-optimales; cependant, nous ne les étudions pas dans le présent document; comme nous donnons des primes en dollars pour les qualités et des coûts en dollars pour les pénalités, nous pouvons approximer la fonction multi-objectif (5.1) - (5.4) par la fonction mono-objectif (C.1) ci-dessous:

$$\min\Big(\sum\nolimits_{j}\big(P_{j} * w_{res}\big) - \sum\nolimits_{l}(Q_{l} * w_{serv}) + \sum\nolimits_{i}\big(P_{down,i} * w_{vm}\big) + \sum\nolimits_{i}(P_{trans,i} * w_{vm})\Big) \tag{C.1}$$

Notre heuristique TS trouve des solutions quasi-optimales pour le problème SARTLM présenté ci-dessus qui est NP-difficile; notre heuristique TS utilise le modèle mathématique ayant la fonction mono-objectif (C.1) sous les contraintes (5.5) - (5.22); aussi, cette modélisation est considérée comme une relaxation MIP-R du modèle mathématique MIP du cinquième chapitre; ce MIP-R nous

permet d’avoir des bornes inferieures pour les résultats de notre heuristique TS; les paramètres de TS et MIP-R sont décrits respectivement au Tableau C.1 et Figure C.1.

Notre heuristique de recherche taboue (TS) utilise une solution initiale comme base de la recherche de solutions; la solution initiale de notre TS est basée sur l’algorithme FFD, et elle est obtenue par le tri décroissant des machines physiques (PMs) selon leurs ressources disponibles et par le tri décroissant des machines virtuelles (VMs) selon leur utilisation des ressources; FFD prend les machines virtuelles (du plus haut au plus bas de l’utilisation des ressources) et les affecte aux PMs (du plus haut au plus bas de la disponibilité des ressources) tout en respectant les contraintes (5.5) - (5.22); quand une VM est attribuée à une PM, FFD met à jour les disponibilités en ressources de cette PM; après avoir trouvé la solution initiale avec l'algorithme FFD, notre heuristique lui applique quelques mouvements pour vérifier s'il y a une meilleure solution dans son voisinage; pour chaque meilleure solution trouvée dans le voisinage, notre TS applique d'autres mouvements sur la meilleure solution courante pour obtenir son voisinage et trouver une autre meilleure solution dans ce voisinage; nombreux mouvements peuvent être appliqués sur une solution par exemple en interchangeant les PMs hébergeant des machines virtuelles ou en déplaçant des machines virtuelles d'une PM vers une autre.

Nous testons notre TS et notre MIP-R avec un simulateur développé en C# (Framework .Net 4.5) avec Windows installé sur PC ayant Intel (R) Core I7 4 GHz et 16 GB de mémoire; notre simulateur reproduit les caractéristiques des requêtes et des machines physiques des traces Google [95]; les caractéristiques des traces Google sont détaillées dans [151]; chaque exécution est répétée 100 fois afin d’avoir les moyennes des résultats; nous utilisons notre propre simulateur plutôt que d'utiliser CloudSim [25] parce que ce dernier n’est pas très approprié pour les migration en temps réel des machines virtuelles selon Pakize et al. [168]; de nombreux tests sont exécutés avec différents échantillons de VMs afin de comparer les résultats de notre TS avec ceux du modèle mathématique "MIP-R" résolu avec CPLEX; nos expériences sont exécutées sur quarante-huit ensembles générés de façon aléatoire à partir des traces Google pour tester nos solutions du problème SARTLM NP-difficile; afin d’interpréter et analyser les résultats, nous calculons les moyennes des pénalités, qualités, temps d’arrêts, et délais de transfert de notre heuristique TS et de notre modèle MIP-R; ces calculs nous permettent de calculer le bénéfice net grâce à la fonction objectif (C.1).

Nous utilisons la programmation en nombres entiers mixte (MIP-R) avec CPLEX pour avoir des bornes inférieures de comparaison pour notre TS; cependant, nous obtenons des erreurs de "dépassement de mémoire" lors de l'essai de MIP-R sur des grands ensembles de VMs à cause des contraintes des interdépendances entre les VMs, en particulier pour des dizaines de milliers de machines virtuelles; MIP-R est définie par la fonction mono-objectif (C.1) avec les contraintes (5.5) - (5.22); notre modèle multi-objectifs du cinquième chapitre est réduit à ce modèle MIP-R mono-objectif à travers la méthode d'agrégation de la somme pondérée de chacun de nos objectifs; avec cette réduction à un modèle mono-objectif, nous trouvons des solutions approximatives pour nos quatre objectifs qui sont les délais de migration des VMs, leur temps d'arrêt, leurs pénalités sur les SLAs, et leur qualité globale; notre modèle proposé permet une flexibilité en affectant un niveau d'importance relative pour chaque objectif; Comme nous ne pouvons pas utiliser MIP-R pour des grands ensembles de VMs, nous utilisons l'algorithme FFD (décrit auparavant) comme une deuxième base de comparaison pour notre heuristique TS; pour comparer les résultats de notre TS, nous calculons aussi les délais CPU de nos algorithmes TS, MIP-R, et FFD; nous calculons aussi le bénéfice net en utilisant la fonction objectif (C.1).

Tableau C.1: Paramètres de notre heuristique

Paramètre	Valeur
$max\ Iterations$:	10.
$max\ Neighbours$:	Nombre de VMs.

```
#CHANGE THE SOLVER (optional)
option solver cplex;
#option solver cplex1251;
#option cplex_options 'timing 1';
#option log_file 'mphapp2.out';
option log_file '.\multi';
#option cplex_options 'mipdisplay 2 mipinterval 1000000';
option solution_precision 10;
#option solver_msg 0;
option version;
option presolve 1;
#option cplex_options 'iisfind 1';
#option solver bonmin;
#option cplex_options 'mipdisplay=2';
option show_stats 1;
option cplex_options 'mipdisplay 2 mipinterval 1';
```

Figure C.1: Paramètres CPLEX de notre MIP-R

La composition de nos scénarios de test est définie au Tableau C.2; elles sont exécutées à la fois par notre heuristique et notre modèle mathématique (MIP-R) (résolus avec CPLEX); les résultats sont détaillés au Tableau C.3 et Tableau C.4; ces tableaux montrent que notre heuristique trouve des solutions rapides et quasi-optimales pour notre problème SARTLM NP-difficile; ces résultats montrent que notre heuristique produit des solutions qui sont en moyenne à 2,55% et au maximum à 7,69% des bornes inférieures obtenues par MIP-R; les temps CPU de notre heuristique sont acceptables et ils ne sont qu'entre 7 secondes et 4314 secondes au-dessus des temps CPU de MIP-R (voir Tableau C.4); par conséquent, notre heuristique est un bon outil pour résoudre le problème SARTLM NP-difficile à résoudre pour de grandes ensembles de VMs interdépendantes.

Nous appliquons plusieurs autres tests avec de plus grands ensembles de VMs (avec des milliers de machines virtuelles), et nous présentons nos résultats au Tableau C.5 et Tableau C.6.

Nous obtenons des erreurs de "dépassement de mémoire" lorsque nous essayons de résoudre, avec MIP-R, notre problème SARTLM avec des grands ensembles de VMs (avec des milliers de machines virtuelles); notre heuristique, vérifiée par les résultats du Tableau C.3 et Tableau C.4, donne de meilleurs résultats que FFD avec des délais raisonnables (voir Tableau C.5 et Tableau C.6); ces tableaux contiennent les résultats de nos nouveaux scénarios (avec des milliers de VMs) du problème SARTLM NP-difficile; chaque nouveau scénario est 1000 fois plus grand que son scenario correspondant détaillé au Tableau C.2 et il est généré aléatoirement à partir des configurations des traces Google; les résultats obtenus avec notre heuristique sont meilleurs que les résultats obtenus avec FFD; nous utilisons FFD comme base de comparaison puisque nous obtenons des erreurs de «dépassement de mémoire» avec MIP-R (résolu par CPLEX) lorsqu'il est appliqué sur des dizaines de milliers de machines virtuelles; les résultats de notre heuristique sont meilleures que ceux de FFD et ils sont obtenus dans des délais acceptables; les solutions de notre heuristique sont meilleures en moyenne de 64%, au minimum de 1%, et au maximum de 194% des solutions de FFD (voir Tableau C.5); les délais CPU de notre heuristique sont en moyenne à 11225, au maximum à 47284, et au minimum à 143 secondes des délais des solutions de FFD (voir Tableau C.6); par conséquent, notre heuristique est un excellent outil pour la résolution du problème SARTLM NP-difficile surtout pour des grands ensembles de VMs interdépendantes. En résumé, les statistiques clés de nos résultats montrent que les solutions de notre TS sont quasi-optimales; ils sont proches des bornes inférieures trouvées grâce à notre modèle mathématique MIP-R, et ils sont meilleurs que ceux de FFD (voir Tableau C.7).

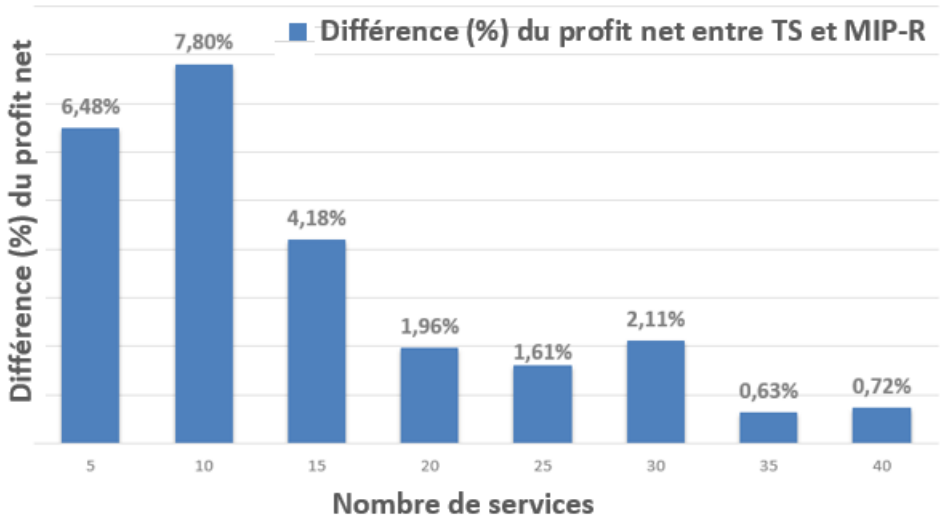

Figure C.2: Écart moyen (en %) entre le bénéfice net de notre heuristique et de MIP-R

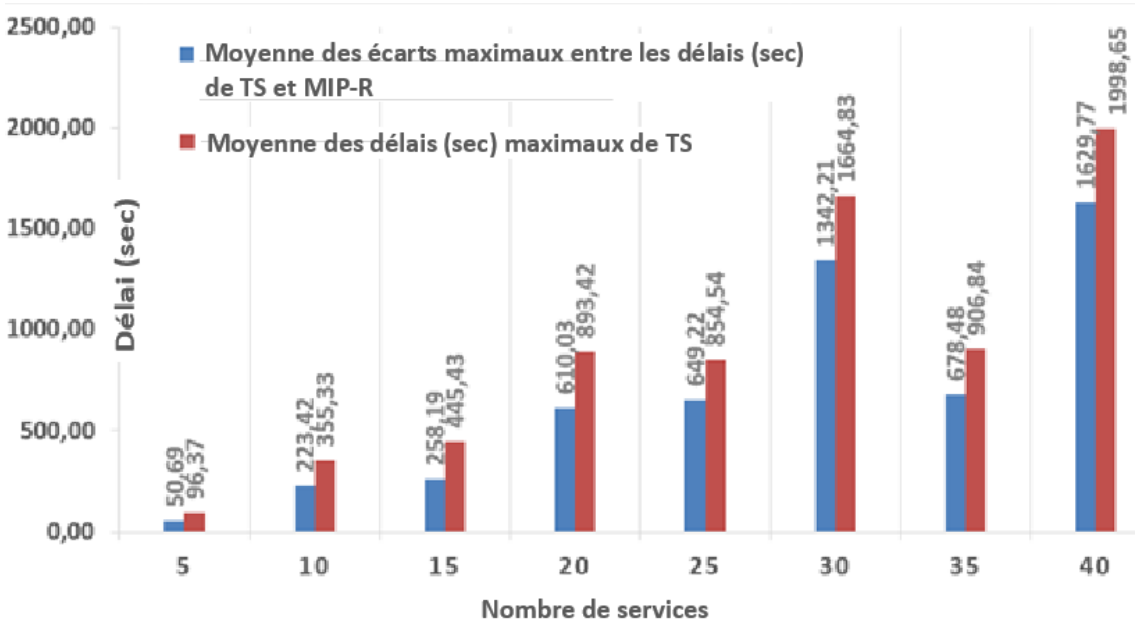

Figure C.3: Moyennes des temps CPU maximums de MIP-R et de notre heuristique

Tableau C.2: Description de nos quarante-huit scénarios de test du problème SARTLM

#Test	|SRVs|	|VMs|	|PMs|
1	5	10	5
2	5	10	10
3	5	20	5
4	5	20	10
5	5	30	5
6	5	30	10
7	10	10	5
8	10	10	10
9	10	20	5
10	10	20	10
11	10	30	5
12	10	30	10
13	15	10	5
14	15	10	10
15	15	20	5
16	15	20	10
17	15	30	5
18	15	30	10
19	20	10	5
20	20	10	10
21	20	20	5
22	20	20	10
23	20	30	5
24	20	30	10
25	25	10	5
26	25	10	10
27	25	20	5
28	25	20	10
29	25	30	5
30	25	30	10
31	30	10	5
32	30	10	10
33	30	20	5
34	30	20	10
35	30	30	5
36	30	30	10
37	35	10	5
38	35	10	10
39	35	20	5
40	35	20	10
41	35	30	5
42	35	30	10
43	40	10	5
44	40	10	10
45	40	20	5
46	40	20	10
47	40	30	5
48	40	30	10

Tableau C.3: Comparaison des coûts de notre TS et MIP-R des quarante-huit scénarios du problème SARTLM

#Test	TS Profit Net ($)			MIP-R Profit Net ($)	Écart (%) TS vs MIP-R Profit Net (%)		
	Max	Moy	Min	Valeur exacte	Max	Moy	Min
1	34	33	32	35	-2,86%	-5,71%	-8,57%
2	21	20	19	22	-4,55%	-9,09%	-13,64%
3	36	36	36	38	-5,26%	-5,26%	-5,26%
4	33	33	33	34	-2,94%	-2,94%	-2,94%
5	53	53	53	55	-3,64%	-3,64%	-3,64%
6	61	60	59	62	-1,61%	-3,23%	-4,84%
7	28	28	28	30	-6,67%	-6,67%	-6,67%
8	12	11	10	13	-7,69%	-15,38%	-23,08%
9	28	28	28	29	-3,45%	-3,45%	-3,45%
10	27	27	27	29	-6,90%	-6,90%	-6,90%
11	55	55	55	57	-3,51%	-3,51%	-3,51%
12	61	61	61	63	-3,17%	-3,17%	-3,17%
13	21	20	19	22	-4,55%	-9,09%	-13,64%
14	22	22	22	23	-4,35%	-4,35%	-4,35%
15	61	61	61	62	-1,61%	-1,61%	-1,61%
16	115	114	113	116	-0,86%	-1,72%	-2,59%
17	149	149	149	151	-1,32%	-1,32%	-1,32%
18	189	188	187	190	-0,53%	-1,05%	-1,58%
19	82	81	80	83	-1,20%	-2,41%	-3,61%
20	82	81	80	83	-1,20%	-2,41%	-3,61%
21	143	143	143	144	-0,69%	-0,69%	-0,69%
22	132	132	132	134	-1,49%	-1,49%	-1,49%
23	242	242	242	244	-0,82%	-0,82%	-0,82%
24	193	192	191	194	-0,52%	-1,03%	-1,55%
25	100	99	98	101	-0,99%	-1,98%	-2,97%
26	96	96	96	97	-1,03%	-1,03%	-1,03%
27	77	77	77	79	-2,53%	-2,53%	-2,53%
28	97	97	97	99	-2,02%	-2,02%	-2,02%
29	350	349	348	351	-0,28%	-0,57%	-0,85%
30	440	440	440	441	-0,23%	-0,23%	-0,23%
31	50	50	50	51	-1,96%	-1,96%	-1,96%
32	45	45	45	46	-2,17%	-2,17%	-2,17%
33	82	81	80	83	-1,20%	-2,41%	-3,61%
34	181	180	179	182	-0,55%	-1,10%	-1,65%
35	187	186	185	188	-0,53%	-1,06%	-1,60%
36	181	180	179	182	-0,55%	-1,10%	-1,65%
37	89	89	89	90	-1,11%	-1,11%	-1,11%
38	118	118	118	119	-0,84%	-0,84%	-0,84%
39	274	274	274	275	-0,36%	-0,36%	-0,36%
40	446	445	444	447	-0,22%	-0,45%	-0,67%
41	610	610	610	612	-0,33%	-0,33%	-0,33%
42	416	416	416	418	-0,48%	-0,48%	-0,48%
43	145	145	145	147	-1,36%	-1,36%	-1,36%
44	142	142	142	143	-0,70%	-0,70%	-0,70%
45	521	520	519	522	-0,19%	-0,38%	-0,57%
46	261	261	261	262	-0,38%	-0,38%	-0,38%
47	336	336	336	338	-0,59%	-0,59%	-0,59%
48	427	426	425	428	-0,23%	-0,47%	-0,70%
Statistiques	**610**	**157**	**10**		**-7,69%**	**-2,55%**	**-0,23%**

Tableau C.4: Comparaison des délais de notre TS et MIP-R des quarante-huit scénarios du problème SARTLM

#Test	TS - Délai (sec)			MIP-R	Écart de TS vs MIP-R Délai (sec)		
	Min	Moy	Max	Délai (sec)	Min	Moy	Max
1	61,67	61,70	61,94	23,73	37,94	37,97	38,21
2	34,70	34,78	35,02	26,86	7,84	7,92	8,16
3	43,13	43,61	43,54	36,06	7,07	7,55	7,48
4	103,25	103,49	103,41	55,31	47,94	48,18	48,10
5	85,01	85,10	85,19	70,12	14,89	14,98	15,07
6	248,27	249,32	249,10	61,99	186,28	187,33	187,11
7	94,25	94,36	96,22	52,75	41,50	41,61	43,47
8	317,34	333,92	338,47	53,73	263,61	280,19	284,74
9	127,94	128,27	129,83	115,34	12,59	12,93	14,48
10	656,87	660,47	661,81	117,41	539,47	543,06	544,40
11	659,03	660,78	665,03	243,72	415,31	417,06	421,31
12	224,79	224,99	240,62	208,52	16,27	16,47	32,10
13	157,38	159,12	173,41	83,70	73,68	75,42	89,71
14	270,01	270,09	278,84	101,96	168,04	168,12	176,88
15	218,20	218,23	218,84	197,86	20,34	20,37	20,98
16	166,49	166,51	177,07	129,77	36,71	36,74	47,30
17	546,17	545,04	550,99	290,70	255,47	254,35	260,29
18	1268,42	1268,51	1273,43	319,43	949,00	949,08	954,00
19	955,88	961,67	961,86	103,08	852,80	858,59	858,79
20	424,88	425,89	426,51	130,72	294,16	295,17	295,79
21	479,01	498,05	523,99	263,21	215,80	234,84	260,78
22	435,02	448,06	448,82	300,81	134,20	147,25	148,01
23	1028,17	1059,54	1091,87	454,02	574,15	605,52	637,86
24	1863,92	1884,04	1907,44	448,49	1415,43	1435,55	1458,95
25	1244,05	1248,40	1248,51	187,00	1057,05	1061,40	1061,51
26	1014,43	1029,95	1044,87	127,94	886,49	902,01	916,93
27	998,64	1002,21	1015,06	184,14	814,50	818,08	830,92
28	422,14	422,42	422,89	253,89	168,25	168,53	169,00
29	459,71	460,06	461,68	246,46	213,25	213,60	215,22
30	926,30	927,38	934,20	232,49	693,81	694,89	701,71
31	2145,98	2152,62	2160,27	144,89	2001,08	2007,72	2015,37
32	589,11	593,45	593,61	131,41	457,70	462,03	462,20
33	724,51	751,43	1072,08	316,60	407,91	434,83	755,48
34	3788,33	4691,01	4771,72	456,90	3331,43	4234,12	4314,83
35	1017,16	1031,59	1054,58	615,61	401,56	415,98	438,97
36	327,49	335,40	336,74	270,32	57,16	65,08	66,42
37	932,25	946,25	967,69	140,22	792,02	806,02	827,46
38	633,16	638,20	648,01	115,33	517,83	522,87	532,68
39	1436,84	1445,84	1468,50	413,13	1023,71	1032,71	1055,37
40	321,17	334,69	337,32	139,50	181,67	195,19	197,82
41	472,43	474,86	479,63	284,67	187,76	190,18	194,96
42	1524,27	1527,19	1539,86	277,29	1246,99	1249,91	1262,57
43	412,73	419,96	477,36	111,04	301,69	308,92	366,31
44	457,83	464,11	478,86	156,82	301,01	307,29	322,04
45	4236,04	4425,08	4426,61	518,46	3717,58	3906,62	3908,15
46	1471,00	1562,97	1572,87	396,27	1074,74	1166,70	1176,60
47	1286,62	1302,91	1309,53	505,81	780,82	797,10	803,72
48	3658,70	3680,87	3726,65	524,88	3133,82	3156,00	3201,77
Statistiques	**34,70**	**884,47**	**4771,72**		**7,07**	**662,79**	**4314,83**

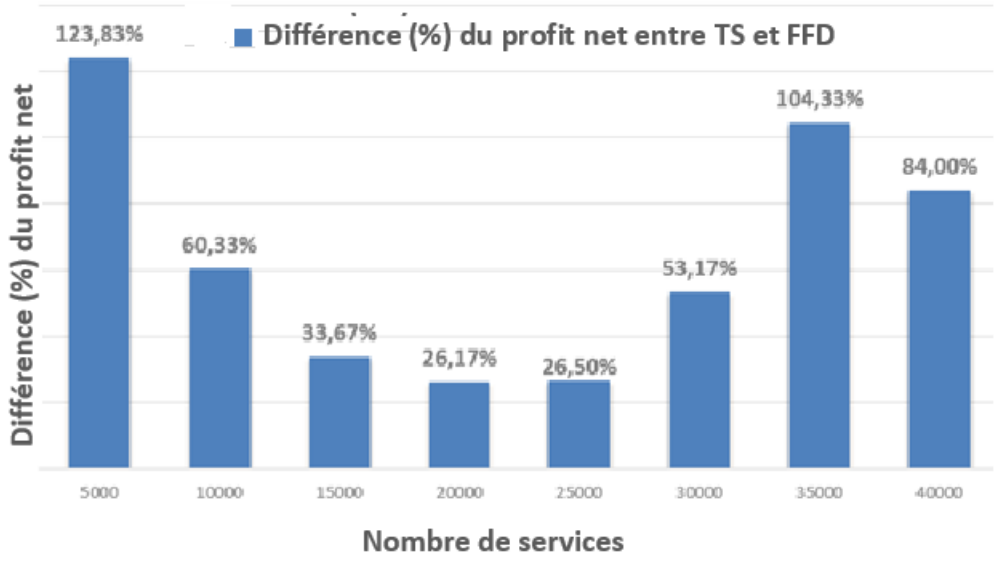

Figure C.4: Écart du bénéfice net (en%) entre notre TS et FFD

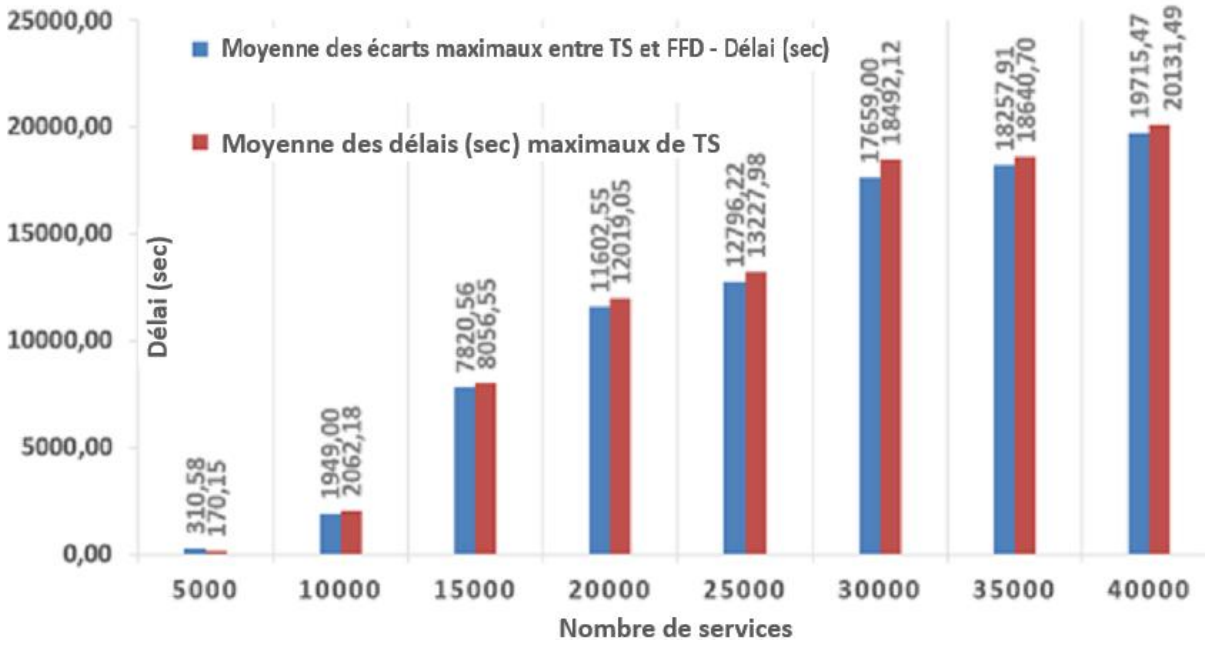

Figure C.5: Moyennes des temps CPU maximums de notre TS

Tableau C.5: Bénéfice net de notre TS pour les grands scénarios du problème SARTLM

#Test	TS Profit Net ($)			FFD Profit Net ($)	Écart (%) TS vs FFD Profit Net		
	Min	Moy	Max		Min	Moy	Max
1 avec 5000 SRVs	3962	3992	4012	1370	189%	191%	193%
2 avec 5000 SRVs	4389	4409	4429	2024	117%	118%	119%
3 avec 5000 SRVs	10810	10810	10810	6160	75%	75%	75%
4 avec 5000 SRVs	2918	2921	2926	991	194%	195%	195%
5 avec 5000 SRVs	14993	14993	14993	10053	49%	49%	49%
6 avec 5000 SRVs	3391	3411	3431	1618	110%	111%	112%
7 avec 10000 SRVs	4410	4410	4410	2564	72%	72%	72%
8 avec 10000 SRVs	10842	10872	10892	6204	75%	75%	76%
9 avec 10000 SRVs	3261	3264	3269	2064	58%	58%	58%
10 avec 10000 SRVs	10786	10786	10786	6548	65%	65%	65%
11 avec 10000 SRVs	4196	4196	4196	2565	64%	64%	64%
12 avec 10000 SRVs	16648	16648	16648	13109	27%	27%	27%
13 avec 15000 SRVs	20418	20448	20468	14205	44%	44%	44%
14 avec 15000 SRVs	27607	27610	27615	22453	23%	23%	23%
15 avec 15000 SRVs	28536	28539	28544	21796	31%	31%	31%
16 avec 15000 SRVs	28688	28708	28728	22136	30%	30%	30%
17 avec 15000 SRVs	32211	32211	32211	26181	23%	23%	23%
18 avec 15000 SRVs	19305	19325	19345	12833	50%	51%	51%
19 avec 20000 SRVs	20670	20700	20720	14176	46%	46%	46%
20 avec 20000 SRVs	27970	27990	28010	21660	29%	29%	29%
21 avec 20000 SRVs	26923	26926	26931	21633	24%	24%	24%
22 avec 20000 SRVs	27109	27109	27109	21122	28%	28%	28%
23 avec 20000 SRVs	5252	5252	5252	4659	13%	13%	13%
24 avec 20000 SRVs	4780	4800	4820	4106	16%	17%	17%
25 avec 25000 SRVs	3625	3645	3665	3586	1%	2%	2%
26 avec 25000 SRVs	3853	3856	3861	3917	2%	2%	1%
27 avec 25000 SRVs	4491	4491	4491	3718	21%	21%	21%
28 avec 25000 SRVs	10841	10841	10841	8060	35%	35%	35%
29 avec 25000 SRVs	7206	7236	7256	4598	57%	57%	58%
30 avec 25000 SRVs	7599	7602	7607	5357	42%	42%	42%
31 avec 30000 SRVs	2303	2306	2311	2727	16%	15%	15%
32 avec 30000 SRVs	13140	13143	13148	9541	38%	38%	38%
33 avec 30000 SRVs	6241	6261	6281	2632	137%	138%	139%
34 avec 30000 SRVs	10829	10849	10869	7506	44%	45%	45%
35 avec 30000 SRVs	8803	8823	8843	5743	53%	54%	54%
36 avec 30000 SRVs	42086	42106	42126	32802	28%	28%	28%
37 avec 35000 SRVs	28401	28404	28409	19782	44%	44%	44%
38 avec 35000 SRVs	23570	23573	23578	17584	34%	34%	34%
39 avec 35000 SRVs	11952	11955	11960	6035	98%	98%	98%
40 avec 35000 SRVs	5540	5570	5590	2396	131%	132%	133%
41 avec 35000 SRVs	4686	4686	4686	1869	151%	151%	151%
42 avec 35000 SRVs	4724	4724	4724	1775	166%	166%	166%
43 avec 40000 SRVs	11959	11959	11959	6330	89%	89%	89%
44 avec 40000 SRVs	6017	6020	6025	3450	74%	74%	75%
45 avec 40000 SRVs	4834	4864	4884	2040	137%	138%	139%
46 avec 40000 SRVs	8448	8451	8456	6378	32%	33%	33%
47 avec 40000 SRVs	4341	4341	4341	1849	135%	135%	135%
48 avec 40000 SRVs	18851	18881	18901	14195	33%	33%	33%
Statistiques	**2303**	**12811**	**42126**		**1%**	**64%**	**194%**

Tableau C.6: Délais CPU de notre TS pour les grands scénarios du problème SARTLM

#Test	TS			FFD	Écart TS vs FFD		
	Délai (sec)			Délai (sec)	Délai (sec)		
	Min	Moy	Max		Min	Moy	Max
1 avec 5000 SRVs	175,561	185,561004	245,561	32,15322061	143,41	153,41	213,41
2 avec 5000 SRVs	238,2304	258,230424	318,23042	31,56261383	206,67	226,67	286,67
3 avec 5000 SRVs	241,1908	281,190759	321,19076	29,23682478	211,95	251,95	291,95
4 avec 5000 SRVs	348,2387	350,238746	360,23875	28,73435094	319,50	321,50	331,50
5 avec 5000 SRVs	305,0457	345,045726	385,04573	26,98202133	278,06	318,06	358,06
6 avec 5000 SRVs	329,2333	349,233329	409,23333	27,3347328	301,90	321,90	381,90
7 avec 10000 SRVs	657,2881	657,288074	657,28807	62,08607948	595,20	595,20	595,20
8 avec 10000 SRVs	1751,83	1761,83003	1821,83	124,5745663	1627,26	1637,26	1697,26
9 avec 10000 SRVs	1777,595	1779,59462	1789,5946	117,5680547	1660,03	1662,03	1672,03
10 avec 10000 SRVs	2177,637	2217,6366	2257,6366	117,993839	2059,64	2099,64	2139,64
11 avec 10000 SRVs	2723,208	2763,20756	2803,2076	135,1697346	2588,04	2628,04	2668,04
12 avec 10000 SRVs	2963,525	3003,52539	3043,5254	121,7010168	2841,82	2881,82	2921,82
13 avec 15000 SRVs	852,2061	862,206083	922,20608	84,48945008	767,72	777,72	837,72
14 avec 15000 SRVs	1081,891	1083,8907	1093,8907	77,51588831	1004,37	1006,37	1016,37
15 avec 15000 SRVs	15037,57	15039,5688	15049,569	294,1589862	14743,41	14745,41	14755,41
16 avec 15000 SRVs	9052,66	9072,65988	9132,6599	329,4989343	8723,16	8743,16	8803,16
17 avec 15000 SRVs	11341,17	11381,165	11421,165	339,6120567	11001,55	11041,55	11081,55
18 avec 15000 SRVs	10639,84	10659,8392	10719,839	290,6876441	10349,15	10369,15	10429,15
19 avec 20000 SRVs	999,9119	1009,91189	1069,9119	99,0138268	900,90	910,90	970,90
20 avec 20000 SRVs	1218,681	1238,68087	1298,6809	89,21951887	1129,46	1149,46	1209,46
21 avec 20000 SRVs	16932,41	16934,4052	16944,405	580,519918	16351,89	16353,89	16363,89
22 avec 20000 SRVs	22409,43	22449,4314	22489,431	637,835017	21771,60	21811,60	21851,60
23 avec 20000 SRVs	21209,77	21249,7699	21289,77	540,6772423	20669,09	20709,09	20749,09
24 avec 20000 SRVs	8942,087	8962,08725	9022,0872	551,6934655	8390,39	8410,39	8470,39
25 avec 25000 SRVs	1114,917	1134,91678	1194,9168	103,5717673	1011,35	1031,35	1091,35
26 avec 25000 SRVs	1562,065	1564,06535	1574,0653	117,0459576	1445,02	1447,02	1457,02
27 avec 25000 SRVs	9058,172	9098,17207	9138,1721	297,2188399	8760,95	8800,95	8840,95
28 avec 25000 SRVs	10671,17	10711,1675	10751,168	296,5755545	10374,59	10414,59	10454,59
29 avec 25000 SRVs	16596,61	16606,608	16666,608	989,2055314	15607,40	15617,40	15677,40
30 avec 25000 SRVs	40030,94	40032,9433	40042,943	786,9317409	39244,01	39246,01	39256,01
31 avec 30000 SRVs	1049,491	1051,49149	1061,4915	102,7196969	946,77	948,77	958,77
32 avec 30000 SRVs	1780,526	1782,52619	1792,5262	218,0820241	1562,44	1564,44	1574,44
33 avec 30000 SRVs	9231,551	9251,55121	9311,5512	324,9869384	8906,56	8926,56	8986,56
34 avec 30000 SRVs	10017,87	10037,8735	10097,873	267,8431783	9750,03	9770,03	9830,03
35 avec 30000 SRVs	37892,03	37912,0331	37972,033	652,5161252	37239,52	37259,52	37319,52
36 avec 30000 SRVs	50637,27	50657,2744	50717,274	3432,62987	47204,64	47224,64	47284,64
37 avec 35000 SRVs	1369,32	1371,31967	1381,3197	133,2869375	1236,03	1238,03	1248,03
38 avec 35000 SRVs	2010,851	2012,85095	2022,8509	141,1341569	1869,72	1871,72	1881,72
39 avec 35000 SRVs	12422,48	12424,4837	12434,484	412,7301255	12009,75	12011,75	12021,75
40 avec 35000 SRVs	12924,2	12934,1996	12994,2	319,0511719	12605,15	12615,15	12675,15
41 avec 35000 SRVs	41320,34	41360,3399	41400,34	695,2223308	40625,12	40665,12	40705,12
42 avec 35000 SRVs	41530,99	41570,9913	41610,991	595,3207593	40935,67	40975,67	41015,67
43 avec 40000 SRVs	1545,195	1585,19488	1625,1949	158,8337951	1386,36	1426,36	1466,36
44 avec 40000 SRVs	1762,841	1764,8408	1774,8408	130,133748	1632,71	1634,71	1644,71
45 avec 40000 SRVs	10981,59	10991,5907	11051,591	356,6892644	10624,90	10634,90	10694,90
46 avec 40000 SRVs	14858,67	14860,6657	14870,666	439,5106132	14419,16	14421,16	14431,16
47 avec 40000 SRVs	45542,54	45582,5388	45622,539	805,5397292	44737,00	44777,00	44817,00
48 avec 40000 SRVs	45774,09	45784,0917	45844,092	605,4148467	45168,68	45178,68	45238,68
Statistiques	**175,56**	**11582,92**	**50717,27**		**143,41**	**11225,58**	**47284,64**

Tableau C.7: Résumé des statistiques clés de nos résultats avec TS

	Différence entre TS et MIP-R		Différence entre TS et FFD	
	Profit Net (%)	**Délai (sec)**	**Profit Net (%)**	**Délai (en sec)**
Moyenne	-2,55%	675,81	63,81%	11225,58
Écart type	2,88%	924,53	49,51%	14239,76
Taille	48	48,00	48	48,00
Coefficient de confiance	1,96	1,96	1,96	1,96
Marge d'erreur	0,82%	261,55	14,01%	4028,45
Borne supérieure	-2,14%	809,25	70,96%	13280,91
Borne inferieure	-2,97%	542,37	56,67%	9170,24
Max	-0,23%	4234,12	195,00%	47224,64
Min	-15,38%	7,55	2,00%	153,41
Marge (Range)	15,15%	4226,57	193,00%	47071,23